

ANONYMOUS PUBLISH-SUBSCRIBE OVERLAYS

JÖRG DAUBERT

vom Fachbereich Informatik
der Technischen Universität Darmstadt

zur Erlangung des Grades
eines Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertationsschrift in englischer Sprache
von Jörg Daubert M.Sc
aus Darmstadt
geboren in Bad Soden-Salmünster

Erstreferent: Prof. Dr. Max Mühlhäuser
Korreferent: Prof. Dr. Thorsten Strufe

Tag der Einreichung: 26. Januar 2016
Tag der Prüfung: 10. März 2016



Fachgebiet Telekooperation
Fachbereich Informatik
Technische Universität Darmstadt
Hochschulkennziffer D-17

Darmstadt 2016

“I grew up with the understanding that the world I lived in was one where people enjoyed a sort of freedom to communicate with each other in privacy without it being monitored, without it being measured, or analyzed or sort of judged by these shadowy figures or systems anytime they mention anything that travels across public lines.”

— Edward Snowden, excerpt from an interview with Glenn Greenwald, 2013

EXECUTIVE SUMMARY

Freedom of speech is a core value of our society. While it can be exercised anonymously towards undesired observers in the physical world, the Internet is based on unique and nonanonymous identifiers (IDs) for every participant. Anonymity, however, is a crucial requirement to exercise freedom of speech using the Internet without having to face political persecution. To achieve anonymity, messages must be unlinkable to senders and receivers. That means that messages cannot be linked to IDs and other identifying information of senders and receivers. Anonymization services, such as Tor, re-establish anonymity within the Internet such that, for example, web content can be consumed anonymously.

Nevertheless, this type of solution embodies two challenges: First, with the appearance of social media, the Internet usage behavior changed drastically from a *one producer with many consumers* to a *many producers with many consumers* of content paradigm. Second, a social media website that is used by many producers and many consumers constitutes a single point of failure (SPoF) regarding both availability and anonymity. Such a website may collect producer and consumer profiles, ultimately breaking anonymity.

Publish/subscribe (pub/sub) is a message dissemination paradigm well suited to address the first challenge, the many-to-many exchange of content. peer-to-peer (P2P) Pub/Sub eliminates the need for an SPoF and, thus, partially addresses the second challenge as well. However, research addressing anonymity as a security requirement for Pub/Sub has merely scratched the surface.

This thesis improves the state-of-the-art in anonymous Pub/Sub in several areas. In particular, the thesis addresses the following aspects of constructing anonymous Pub/Sub systems: (i.) Building blocks that reduce the complexity of constructing anonymous Pub/Sub systems are proposed; (ii.) methods for anonymously establishing Pub/Sub overlay networks are presented; (iii.) a method for inter-overlay optimization to distribute load is introduced; (iv.) methods for optimizing overlays regarding anonymity are proposed, and (v.) anonymity attacks and countermeasures are presented.

CONTRIBUTIONS This thesis contributes to the following research categories:

ANONYMOUS OVERLAY ESTABLISHMENT An anonymous Pub/Sub system is presented along six self-containing building blocks with the goal of establishing overlay networks that transport notifications from publishers to subscribers. Each building block is discussed in detail with a focus on leveraging related work to realize the building block. For *attribute localization*, the building block most relevant for establishing overlays, this thesis

proposes multiple contributions: the usage of hash chains as a privacy-preserving transaction pseudonym and distance metric; the adaptation of flooding as well as forest fires; and random walks to distribute attribute knowledge.

ANONYMOUS OVERLAY OPTIMIZATION The thesis proposes two optimizations for anonymity and one optimization for balancing the load. The first anonymity optimization, probabilistic forwarding (PF), applies the concept of mimic traffic to the domain of Pub/Sub. The second anonymity optimization, the shell game (SG), shuffles the overlay. Both optimizations prevent an exposure of information to attackers that can gain knowledge about the overlay topology. The load-balancing optimization uses a ring communication and Bloom filters to distribute load among nodes.

ANONYMITY ATTACKS AND COUNTERMEASURES Several well-known anonymity attacks are adapted to the domain of anonymous Pub/Sub and evaluated in detail. Novel attacks, such as the request/response-attack and the corner attack, are proposed as well and complemented with countermeasures.

EVALUATION The proposed mechanisms and attacks are evaluated using a qualitative approach, quantitatively with an extensive simulation, and empirically with a proof of concept (POC) application. The qualitative approach indicates that the presented mechanisms are well-suited to protect anonymity against a malicious insider threat.

The quantitative evaluation is performed with the event-based simulation framework OMNeT++. The results show that the presented anonymous Pub/Sub system can protect anonymity, even in case malicious insiders are combined with a global observer of a very strong anonymity threat. The results also reveal in which situations PF or the SG provides the better anonymity protection. Furthermore, the results indicate which capabilities are favorable for an anonymity attacker.

An anonymous micro-blogging application for Twitter shows that the presented system can be implemented for a real-world use case: With the application, users exchange tweets via hashtag-overlays, and cryptographic keys via quick response (QR)-codes and near field communication (NFC).

ZUSAMMENFASSUNG

Die Redefreiheit zählt zu den wichtigsten Werten unserer Gesellschaft. Während die Redefreiheit in der physischen Welt anonym gegenüber unerwünschten Fremden ausgeübt werden kann, basiert das Internet auf nicht-anonymen, sondern vielmehr eindeutigen, Identifikationsmerkmalen (IDs). Zur Ausübung der Redefreiheit ohne politische Verfolgung ist die Anonymität jedoch eine Kernanforderung. Anonymity bedingt, dass eine Nachricht nicht mit Sender und Empfänger in Verbindung gebracht werden kann. Insbesondere darf eine Nachricht nicht mit IDs oder anderen Identifikationsmerkmalen in Beziehung gesetzt werden.

Anonymisierungsdienste wie Tor können im Internet verwendet werden, um etwa Webinhalte anonym abzurufen. Mit einer solchen Lösung ergeben sich jedoch 2 Arten von Herausforderungen: Erstens hat sich die Internet-Nutzung mit dem Aufkommen sozialer Medien verändert; statt einem *ein Produzent mit vielen Konsumenten* von Inhalten Nutzungs-Paradigma nimmt nun das *viele Produzenten mit vielen Konsumenten*-Paradigma eine gewichtige Rolle ein. Zweitens stellt eine Social-Media-Webseite mit vielen Nutzer eine zentrale Schwachstelle da, sowohl in Bezug auf die Verfügbarkeit als auch auf die Anonymität. Eine solche Webseite kann Profildaten ihrer Nutzer sammeln um schlussendlich deren Anonymität zu brechen.

Publish/subscribe (pub/sub) ist ein Konzept zur Nachrichten-Verteilung, welches die erste Herausforderung sehr gut adressiert. **P2P Pub/Sub** kommt ohne zentrale Schwachstelle aus, und deckt damit auch die zweite Herausforderung teilweise ab. Jedoch steht Forschung, welche Anonymität als Sicherheitsanforderung für **Pub/Sub** berücksichtigt, erst am Anfang.

Diese Dissertation erweitert den aktuellen Forschungsstand in mehreren Bereichen. Die folgenden Aspekte der Konstruktion von anonymen **Pub/Sub** Systemen werden dabei angesprochen: (i.) Bausteine zur Erleichterung zukünftiger Entwicklung von anonymen **Pub/Sub** Systemen werden vorgeschlagen; (ii.) Methoden zur anonymen Erzeugung von Overlay-Netzwerken werden vorgestellt; (iii.) eine anonyme Inter-Overlay-Optimierung zur Verteilung von Last wird präsentiert; (iv.) Methoden zur Optimierung der Anonymität in Overlay-Netzwerken werden eingebührt; (v.) Angriffe der Anonymität sowie Schutzmaßnahmen werden vorgeschlagen.

BEITRÄGE Die Beiträge dieser Dissertation lassen sich in folgende Forschungsschwerpunkte einteilen:

ANONYME OVERLAY-ERZEUGUNG Es wird ein anonymes **Pub/Sub** System anhand von sechs in sich geschlossenen Bausteinen vorgestellt. Hierbei wird das Ziel verfolgt, Overlay-Netzwerke zu erzeugen, welche nachfolgend Nachrichten von Produzenten zu Konsumenten transportieren. Jeder Baustein wird im Detail

und in Bezug auf relevante verwandte Arbeiten erläutert. Für einen der wichtigsten Bausteine, die Lokalisierung von Attributen (*attribute localization*), werden mehrere Beiträge geleistet: Hash-Ketten kombinieren Transaktions-Pseudonym und Distanz-Metrik unter Schutz der Privatsphäre; sogenanntes flooding, forest fires, und doppelte random walks werden zum Verteilen von Attributs-Wissen verwendet.

ANONYME OVERLAY-OPTIMIERUNG Die Dissertation stellt zwei Optimierungen zur Verbesserung der Anonymität sowie eine Optimierung zur Verteilung von Last vor. Die erste Anonymitäts-Optimierung, probabilistic forwarding (PF), überträgt das Konzept von “mimi traffic” auf Pub/Sub. Die zweite Optimierung, das Hütchen-Spiel (shell game (SG)), zerwürfelt Overlay-Netzwerke. Beide Optimierungen verfolgen das Ziel, die Preisgabe von Information durch die Overlay-Topologie zu vermeiden. Die Optimierung zur Last-Verteilung verwendet einen Kommunikations-Ring sowie Bloom-Filter um die Last unter den Knoten von Overlay-Netzwerken zu verteilen

ANONYMITÄTS-ANGRIFFE UND SCHUTZMASSNAHMEN Mehrere bekannten Anonymitäts-Attacken werden auf anonymes Pub/Sub angewendet und im Detail analysiert. Weiterhin werden neue Attacken, wie etwa die Anfrage/Antwort-Attacke (request/response-attack) sowie die Einengungs-Attacke (corner-attack), vorgestellt und mit entsprechenden Schutzmechanismen versehen.

AUSWERTUNG Die vorgestellten Mechanismen und Angriffe werden anhand von drei Analyse-Verfahren ausgewertet: einem qualitativen Ansatz, einer umfangreichen Simulation, sowie empirisch mit einem Prototyp. Der qualitative Ansatz deutet an, dass die vorgestellten Mechanismen gut zum Schutz der Anonymität gegenüber der Gefahr von böartigen Insidern geeignet ist.

Die quantitative Analyse wird mit einer Ereignis-diskreten Simulation anhand des OMNeT++ Werkzeugs durchgeführt. Den Ergebnissen folgend ist das vorgestellte anonyme Pub/Sub System in der Tat zum Schutz der Anonymität geeignet, selbst wenn ein besonders starkes Angreifer-Modell aus böartigem Insider sowie globalen Überwacher angenommen wird. Die Resultate zeigen auch, in welchen Situation PF oder das SG besseren Schutz bieten. Weiterhin ergeben die Resultate, welche Fähigkeiten einem Angreifer am Meisten nutzen.

Für die empirische Auswertung wurde eine anonyme Micro-Blogging Anwendung für Twitter entwickelt. Diese Anwendung zeigt, dass die in dieser Dissertation vorgestellten Mechanismen in der Tat in realistischen Szenarien eingesetzt werden können. Mit der Anwendung können Teilnehmer Tweets über Hashtag-Overlay-Netz austauschen. QR-Codes und NFC dienen der Verteilung von Schlüsselmaterial.

PUBLICATIONS

- Jörg Daubert, Leon Böck, Panayotis Kikiras, Max Mühlhäuser, and Mathias Fischer. „Twitterize: Anonymous Micro-blogging.“ In: *11th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2014, Doha, Qatar, November 10-13, 2014*. IEEE, 2014, pp. 817–823. ISBN: 978-1-4799-7100-8. DOI: [10.1109/AICCSA.2014.7073285](https://doi.org/10.1109/AICCSA.2014.7073285). URL: <http://dx.doi.org/10.1109/AICCSA.2014.7073285>.
- Jörg Daubert, Mathias Fischer, Tim Grube, Stefan Schiffner, Panayotis Kikiras, and Max Mühlhäuser. „AnonPubSub: Anonymous publish-subscribe overlays.“ In: *Elsevier Computer Communications* 76 (2016), pp. 42–53. ISSN: 0140-3664. DOI: [http://dx.doi.org/10.1016/j.comcom.2015.11.004](https://doi.org/10.1016/j.comcom.2015.11.004). URL: <http://www.sciencedirect.com/science/article/pii/S0140366415004211>.
- Jörg Daubert, Mathias Fischer, Stefan Schiffner, and Max Mühlhäuser. „Distributed and Anonymous Publish-Subscribe.“ In: *Network and System Security - 7th International Conference, NSS 2013, Madrid, Spain, June 3-4, 2013. Proceedings*. Ed. by Javier Lopez, Xinyi Huang, and Ravi Sandhu. Vol. 7873. Lecture Notes in Computer Science. Springer, 2013, pp. 685–691. DOI: [10.1007/978-3-642-38631-2_57](https://doi.org/10.1007/978-3-642-38631-2_57). URL: http://dx.doi.org/10.1007/978-3-642-38631-2_57.
- Jörg Daubert, Tim Grube, Max Mühlhäuser, and Mathias Fischer. „Internal attacks in anonymous publish-subscribe P2P overlays.“ In: *2015 International Conference and Workshops on Networked Systems, NetSys 2015, Cottbus, Germany, March 9-12, 2015*. IEEE, 2015, pp. 1–8. ISBN: 978-1-4799-5804-7. DOI: [10.1109/NetSys.2015.7089074](https://doi.org/10.1109/NetSys.2015.7089074). URL: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7083793>.
- Jörg Daubert, Tim Grube, Max Mühlhäuser, and Mathias Fischer. „On the anonymity of privacy-preserving many-to-many communication in the presence of node churn and attacks.“ In: *13th Annual IEEE Consumer Communications and Networking Conference, CCNC 2016, Las Vegas, NV, USA, January 9-12, 2016*. IEEE, 2016, pp. 745–751. ISBN: 978-1-4673-9292-1.
- Jörg Daubert, Alexander Wiesmaier, and Panayotis Kikiras. „A view on privacy & trust in IoT.“ In: *IEEE International Conference on Communication, ICC 2015, London, United Kingdom, June 8-12, 2015, Workshop Proceedings*. IEEE, 2015, pp. 2665–2670. DOI: [10.1109/ICCW.2015.7247581](https://doi.org/10.1109/ICCW.2015.7247581). URL: <http://dx.doi.org/10.1109/ICCW.2015.7247581>.
- Sebastian Funke, Jörg Daubert, Alexander Wiesmaier, Panayotis Kikiras, and Max Mühlhäuser. „End-2-End Privacy Architecture for IoT.“ In: *IEEE Conference on Communications and Network Security*,

CNS 2015, poster proceedings, Florence, Italy, September 28-30, 2015.
IEEE, 2015, pp. 705–706. DOI: [10.1109/CNS.2015.7346895](https://doi.org/10.1109/CNS.2015.7346895).

ACKNOWLEDGMENTS

This thesis would not exist without the support of my advisors, colleagues, friends, and my family. Therefore, I would like to thankfully acknowledge my supporters.

First, a very big thank you to Max Mühlhäuser, who granted me the opportunity to write this thesis and who continuously supported me in getting to this very stage—the completed thesis. I would also like to thank Thorsten Strufe for inspiring this thesis with his research, and for acting as the second advisor.

Also, I have to thank my PostDocs Mathias Fischer, Stefan Schiffner, and Leonardo Martucci for their patience, for their continuous support, for getting me always back on track, and for all those great research discussions. Thank you very much.

And of course, I am delighted to thank all my great colleagues from the Smart Security and Trust (SST) area and the Secure Smart Infrastructures (SSI) area at the Telecooperation (TK) Lab at TU Darmstadt: Fábio Borges de Oliveira, Florian Volk, Sascha Hauke, Sheikh Mahbub Habib, Emmanouil Vasilomanolakis, Shankar Karuppayah, Carlos Garcia Cordero, Tim Grube, and Siavash Valipour. You are awesome!

I also have to thank our visiting guest researchers that always stimulated fruitful discussions: Dan Yamamoto, Hiroki Uchiyama, and Andrea Tundis. A big thank you to Benedikt Schmidt, Immanuel Schweizer, Stefan Radomski, Kai Höver, and all the others for their input and their support.

I would also like to thank the staff at TK for supporting me in all administrative tasks: Elke Halla, Jörg Dähler, Nina Jäger, Karin Tillack, Elke Reimund, Fabian Herrlich, Silke Romero-Ostermann, and Pamela Milojevic.

Also, a big thank you to all the colleagues from AGT International—Joachim Schaper, Rainer Müller, Panayotikis Kikiras, Alexander Wiesmaier, Sebastian Schmerl, Mario Lischka, Peter Ebinger, Richard Gold, Hariharan Rajasekaran, Christian Gorecki, Andy Rupp, Alban Hessler, Sven Schaust, Martin Strohbach, and many more—for offering me this opportunity and for supporting me.

Finally, a great thank you to my friends and family.

CONTENTS

1	INTRODUCTION	1
1.1	Problem statement	1
1.2	Contributions	2
1.3	Structure of this thesis	5
2	BACKGROUND	7
2.1	Publish/Subscribe	7
2.1.1	Concept and terminology	8
2.1.2	Requirements	11
2.2	Anonymity	13
2.2.1	Concepts	14
2.2.2	Anonymization systems	16
2.2.3	Metrics	18
2.3	Anonymization implementations & services	21
2.3.1	anon.penet.fi	21
2.3.2	Cypherpunk remailers	21
2.3.3	Mixminion	22
2.3.4	Crowds	22
2.3.5	Tarzan	22
2.3.6	Tor	23
2.3.7	MorphMix	26
2.3.8	Anonymizer.com	26
2.3.9	JonDo / AN.ON & JAP	26
2.3.10	P5	27
2.3.11	Summary	27
2.4	P2P Systems	27
2.4.1	Structured overlays	28
2.4.2	Unstructured overlays	28
2.5	Network Simulation	29
2.5.1	Discrete event simulation	30
2.5.2	OMNeT++	30
2.5.3	Summary	33
2.6	Summary	33
3	REQUIREMENTS AND STATE OF THE ART	35
3.1	Publish/subscribe system formalization	35
3.2	Anonymity requirements for Pub/Sub	39
3.3	Attacker models for anonymity	41
3.3.1	Properties & capabilities	42
3.3.2	Selected attackers	43
3.4	Attacks on anonymous pub/sub	44
3.4.1	A timing attack in anonymous communication	45
3.5	Structuring system architectures	50
3.6	Building blocks	54
3.6.1	Basic membership	54
3.6.2	Attribute localization	57
3.6.3	Key management	60
3.6.4	Matching	62

3.6.5	Community management	65
3.6.6	Content distribution	66
3.6.7	Guideline for Pub/Sub system construction	68
3.7	Anonymous Pub/Sub systems	69
3.7.1	Development of anonymous Pub/Sub	69
3.7.2	Discussion of anonymous Pub/Sub systems	70
3.7.3	Summary	73
3.8	Summary	74
4	AN ANONYMOUS UNSTRUCTURED PUBLISH/SUBSCRIBE OVERLAY	77
4.1	Basic membership	78
4.2	Key management	79
4.3	Attribute localization	81
4.3.1	Flooding	82
4.3.2	Forest fire	88
4.3.3	Double random walk	91
4.4	Matching	95
4.5	Community management	96
4.5.1	Node churn and overlay connectivity	96
4.5.2	Inter-overlay optimization with Bloom filters	99
4.6	Content distribution	107
4.7	Cover Traffic in Anonymous Pub-Sub	110
4.8	Overlay Randomization in Anonymous Pub-Sub	112
4.8.1	Cornering nodes with the shell game	116
4.9	Summary	117
5	EVALUATION	119
5.1	Request / response-based internal attack	119
5.2	Qualitative properties	122
5.3	Quantitative simulation	124
5.3.1	Simulation	124
5.3.2	Setup	125
5.3.3	Evaluation metrics	126
5.3.4	Basic overlay properties and attack accuracy	129
5.3.5	Forwarding probability and attack accuracy	131
5.3.6	Shell game parameter and attack accuracy	132
5.3.7	Probabilistic forwarding & shell game	135
5.3.8	Request / response-based internal attacks	135
5.3.9	Attacks based on churn and disconnecting nodes	139
5.3.10	Summary of the simulation	140
5.4	Proof-of-concept: Twitterize	141
5.4.1	Privacy-preserving twitter apps	142
5.4.2	Twitterize	143
5.4.3	Evaluation	146
5.4.4	Summary of the POC	147
5.5	Summary	148
6	CONCLUSION	151
6.1	Summary	151
6.2	Future work	154
	BIBLIOGRAPHY	155

LIST OF FIGURES

Figure 1	Basic membership (bottom), Pub/Sub overlay (middle), community (top)	3
Figure 2	Communication network annotated with a probability distribution after one attack step.	4
Figure 3	A simple object-based publish/subscribe system. [51]	8
Figure 4	Methods for disseminating notifications	10
Figure 5	Types of MIX networks (MIX-nets) [45]	16
Figure 6	Illustration of onion routing [168].	17
Figure 7	Anonymity set sizes. [112]	18
Figure 8	Objective Modular Network Testbed in C++ (OMNeT++) simulation GUI. [94]	31
Figure 9	StandardHost compound module of INET, depicted in Network Description (NED) visualization.	32
Figure 10	Example graph with node IDs and probabilities.	48
Figure 11	Anonymization service cannot full the application from exposing information to the attacker.	51
Figure 12	Design considerations—building blocks and interactions. The numbers indicate the lifecycle phases.	53
Figure 13	Building blocks—role considerations	53
Figure 14	EventGuard architecture [146]	71
Figure 15	Base Mechanism for Subscription and Multicast Tree Creation. [126]	73
Figure 16	Basic membership (bottom), Pub/Sub overlay (middle), community (top)	78
Figure 17	Key distribution sequence	81
Figure 18	Transactions pseudonyms for advertisements. Left: loop detection. Right: duplicate detection.	83
Figure 19	Signatures for advertisements: Distribution of trusted third party (TTP) public key (o), request for and response for signature key (1,2) , forwarding and message validation (3,4).	86
Figure 20	Distribution of advertisements. Forwarder increment hash chain elements ($h \rightarrow h' \rightarrow h''$) and merge advertisements when forwarding.	87
Figure 21	Attribute localization with forest fire.	89
Figure 22	Density plot of the Weibull distribution, $\lambda = 2, k = 2$	91
Figure 23	Attribute localization with double random walks. $TTL_p^a = 3$	93

Figure 24	Messages from attribute localization to the left as reference. Step 1: a subscriber compares interest t_x with advertised attribute t_a . Step 2: the subscriber subscribes towards f . Step 3: f forwards the subscription to p	95
Figure 25	Effects of churn.	98
Figure 26	Example with 3 attributes: a_1 in white/solid, a_2 in light gray/dotted, and a_3 in dark gray/-dashed. Node 4 is overloaded.	100
Figure 27	Inter-overlay optimization community: initiator is node 4, nodes 1, . . . , 8 participate.	101
Figure 28	Inter-overlay optimization community connected as a ring with initiator 4.	101
Figure 29	Key exchange and message flow for confidential notifications.	109
Figure 30	Hiding nodes dashed, receivers of cover messages gray. Middle: probabilistic forwarding with one hop of neighbors. Right: two hops of neighbors.	111
Figure 31	Initial state and two steps of the shell game. The active node is marked with a dashed outline, swapped nodes are marked in gray.	114
Figure 32	Corner attack: A changes into a position close to s , disconnects s , and then observes s	117
Figure 33	Frist two attack steps.	121
Figure 34	Third and fourth attack step.	121
Figure 35	StandardHost structure in the INET framework representing a node $v \in V$	125
Figure 36	Overview of the AnonPubSub implementation in OMNeT++ as user datagram protocol (UDP) application.	126
Figure 37	Evaluation of the anonymity attack accuracy.	129
Figure 38	Influence of PF on the anonymity attack by the G \mathcal{A}	131
Figure 39	Overhead caused by PF.	132
Figure 40	Convergence of the SG (1).	133
Figure 41	Convergence of the SG (2).	134
Figure 42	Influence of the basic overlay on the I \mathcal{A}	137
Figure 43	Anonymity attacks by the I \mathcal{A} with collusion.	138
Figure 44	Anonymity attacks performed by I \mathcal{A}	140
Figure 45	Twitter social graph with follow relation and tweet flow. The bold arrows represent the overlay for a hashtag on top of the social graph underlay.	143
Figure 46	Advertisement tweets propagate in Twitter over timelines to followers. If an advertisement is new for a follower, the follower increments the hash element and posts it to her own timeline. Hence, the follower becomes a forwarder.	144

Figure 47	Subscription tweets propagate via direct tweets in Twitter. Users have to main a routing table as state outside of Twitter for the protocol. . .	144
Figure 48	Software design of the Twitterize application. Parts of the design a shown as part of the information flow. From left to right: information flow and design for publishing tweets, forwarding tweets, and receiving tweets. The app es each shown at the top, functions provided by Twitter at the bottom.	145
Figure 49	Tweets per hashtag over the number of hashtags in the system. The system degrades quickly with increasing overlay sizes, e.g., max overlay size. However, with an overlay membership rate of 10%, the system can still handle one tweet per hour for 1,000 hashtags.	148

LIST OF TABLES

Table 1	System formalization	39
Table 2	Comparison of basic membership approaches. The symbols +, o, - indicate how well the approaches perform for the given properties. . .	57
Table 3	Comparison attribute localization approaches	60
Table 4	Comparison key exchange approaches.	62
Table 5	Comparison of matching approaches.	65
Table 6	Comparison of related work. The symbols +, o, - indicate how well the approaches perform for the given properties. EventGuard requires add-on anonymity protection to fulfill the requirements marked as o.	74
Table 7	Simulation parameters	127
Table 8	Simulation metrics	127
Table 9	Default parameter settings for internal attacks.	136
Table 10	Basic membership management in Twitter . .	143
Table 11	Basic membership management in Twitter . .	146
Table 12	Twitter limitations	147

ACRONYMS

API	Application Programming Interface
ABAC	attribute-based access control
ABE	attribute-based encryption
AES	advanced encryption standard
ALM	application layer multicast
BFS	breadth-first search
BL	base line
C-CBPS	confidential content-based publish/subscribe
CBC	cipher-block chaining
CIA	confidentiality integrity availability
DC-net	dining cryptographers networks
DH	Diffie-Hellman
DHT	distributed hash table
DoS	denial of service
DDoS	distributed denial of service
DFS	depth-first search
E2E	end-to-end
F2F	friend-to-friend
FN	false negative
FP	false positive
HMAC	keyed-hash message authentication code
HTTP	hypertext transfer protocol
IBE	identity-based encryption
ID	identifier
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet protocol
IPC	inter-process communication

ISP	Internet service provider
JAP	Java Anon Proxy
JMS	Java message service
MCON	membership-concealing overlay network
MITM	man in the middle
MIX-net	MIX network
MLCE	multiple layer commutative encryption
NAT	network address translation
NED	Network Description
NFC	near field communication
NSA	US National Security Agency
OMNeT++	Objective Modular Network Testbed in C++
OSI	Open Systems Interconnection
OSN	online social network
OT	oblivious transfer
P2P	peer-to-peer
PEKS	public key encryption with keyword search
PET	privacy enhancing technology
PF	probabilistic forwarding
PII	personally identifiable information
PIM	protocol independent multicast
POC	proof of concept
PtP	point-to-point
Pub/Sub	publish/subscribe
QR	quick response
RPC	remote procedure call
RTT	round-trip time
SG	shell game
SHA	secure hash algorithm
SPoF	single point of failure
Tails	The amnesic incognito live system

Tor the onion router

UDP user datagram protocol

TLM transport layer multicast

TLS transport layer security

TTL time to live

TTP trusted third party

VPN virtual private network

XACML eXtensible Access Control Markup Language

ZKP zero-knowledge proof

INTRODUCTION

Many of today's Internet-based applications are centered around the exchange of information *between* users. This information exchange often follows the *many-to-many* paradigm [147], meaning many users consume the same information, whereas many users also contribute to this information. Examples of this paradigm are manifold: in micro-blogging, many users produce articles on the same topic, and many users read the same article. Hashtags in Twitter are a prime example for that: publishers annotate their tweets with hashtags; Twitter streams search results for hashtags to subscribers. Likewise, emerging applications such as private ridesharing [127] and Internet of Things (IoT) applications [124] build upon the same paradigm, i.e., many-to-many communication concerning shared interests. Esposito and Ciampi [49] list many more large-scale Pub/Sub applications that fit the Pub/Sub paradigm. Nevertheless, most of these applications do not use a Pub/Sub system.

Pub/Sub is a form of message dissemination that follows exactly this type of information exchange paradigm. Pub/Sub introduces the notion of a publisher, an entity that produces messages, and a subscriber, an entity that consumes messages. Subscribers articulate their interest in messages via so-called subscriptions. The Pub/Sub system facilitates the message exchange from publisher to subscriber. The two entities are loosely coupled in several ways and are not required to know each other.

Considering the application scenarios such as micro-blogging and the IoT, personal information is exchanged using such Pub/Sub systems. Privacy and anonymity [40] are therefore desirable [65], among other security requirements.

1.1 PROBLEM STATEMENT

As of today, this Pub/Sub functionality is typically realized in a centralized manner by the platform provider of the application. For instance with micro-blogging applications, the message dissemination is often realized by a central cloud service. However, such a central service results in the service provider learning about all messages exchanged between users. This causes a privacy issue, i.e., a user having no privacy towards the service provider. While the lack of privacy might be considered uncomfortable but tolerable for some users, the lack of privacy already leads to negative repercussions for some users, such as those who are blogging under political repression being followed by persecution, for example. Moreover, emerging applications such as the mentioned ridesharing and the IoT, may cause yet unknown consequences without privacy.

Attempts have been made to resolve the privacy issues; mostly via access control [6] and encryption [138]. Both approaches, however, merely protect privacy against undesired users and spying eyes: encryption, without being part of an anonymization service, only protects the confidentiality of information; encryption does not protect metadata, and thus not the anonymity. The service provided with central cloud service remains a SPoF for privacy, and it learns which users produce and consume information. Worse, a compromise of the SPoF cloud service can have terrible [144] consequences [156] for the users, as the recent Ashley Madison data breach [130, 155] revealed once more in a long history of service provider hacks.

Efforts to distribute the SPoF, as well as to route information from publishers to subscribers on encrypted messages, have been made [16]. Still, the flow of messages from publishers to subscribers may be traced by spying eyes. Combined with background information about the popularity of information, privacy can be broken. Anonymous communication [21, 22] is an approach to protect the senders or receivers of messages from such message tracing. Anonymization services used today [44, 47] are designed for point-to-point (PtP) communication rather than many-to-many communication. As a result, these services cause either signaling overhead regarding messages or require a centralized service again to mediate the messages.

This thesis aims to overcome the mentioned challenges by combining distributed Pub/Sub with anonymization services. The following research questions will be answered in this thesis:

- How can a distribution overlay network for Pub/Sub be established without revealing the identities of publishers and subscribers?
- How can these identities be concealed while notifications of arbitrary size and at arbitrary rate are distributed in such an overlay?
- What are realistic anonymity attacker models for such systems, what attacks can be performed, and how can the attack success be measured?
- Which approaches can be used to protect anonymity against these attacks?
- How can an anonymous Pub/Sub system be optimized, e.g., to handle churn and to balance the load, without compromising anonymity?

The solutions to these research questions are summarized in the next section.

1.2 CONTRIBUTIONS

This thesis provides several novel contributions to research questions mentioned above as well as anonymous Pub/Sub in general.

ANONYMOUS P2P PUBLISH/SUBSCRIBE Related privacy-preserving Pub/Sub systems [118, 138, 154] address privacy and the protection of confidentiality against an attacker who possesses no key material and does not actively influence the system. These systems, therefore, cannot protect against malicious insiders [38].

The first contribution to this thesis is a generic model for anonymous Pub/Sub and has been published in [36]. This model serves as a framework for a novel categorization of anonymous Pub/Sub in so-called *building blocks*, each addressing a specific functionality. Existing solutions for the building blocks as well as the generic model are presented in Chapter 3.

The systematic discussion of technologies is followed by the construction of a new anonymous Pub/Sub system in Chapter 3, which has been published in [36, 37]. As shown in Figure 1, the system performs attribute localization—the building block responsible for localization-relevant publications—on top of the basic membership layer. Attribute localization is formed by a building block responsible for establishing connectivity between users.

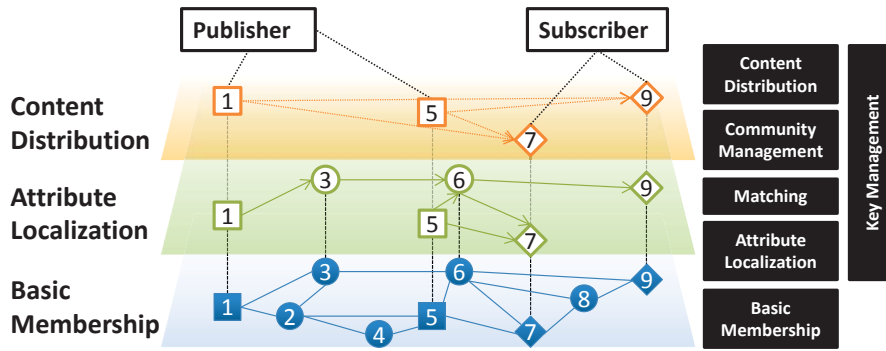


Figure 1: Basic membership (bottom), Pub/Sub overlay (middle), community (top)

The system is then capable of performing content distribution with low latency and without any restrictions regarding message size and rate. This system protects anonymity against malicious insiders, i.e., nodes in the basic membership that possess key material and can generate valid messages. POCs of this system have been published in [35, 59].

ATTACKS AGAINST ANONYMOUS PUB/SUB Anonymization services such as the onion router (Tor) have been analyzed with common anonymity attacker models such as the malicious insider threat [44]. However, such an analysis is usually performed theoretically [5, 43] and thus describes the devastating but only theoretical worst case. Real-world attacks against anonymization have shown that flaws in the attacker model can easily lead to such a worst case in reality [68, 163]. This lead to the issues that (i) the attacker models considered in related work are weak and (ii) real-world assessments are missing.

The second contribution of this thesis is therefore practical attacks in Chapter 5. The request/response-based attack has been published in [38] and can be applied to the anonymous P2P system by collud-

ing malicious insiders. This attack adaptively probes the communication network and evaluates the responses from other nodes. These responses are used to refine a probability distribution as shown in Figure 2 that can be ultimately used to de-anonymize the responders, e.g., subscribers in the case of Pub/Sub.

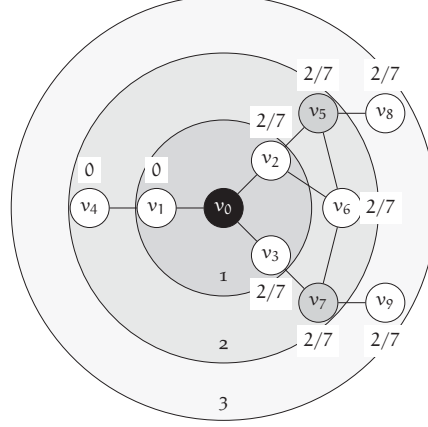


Figure 2: Communication network annotated with a probability distribution after one attack step.

Further attacks contributing to this thesis have been published in [39] and use more invasive methods, such as interrupting the connection to a node in the communication network to monitor the response.

ANONYMITY-ENHANCING TECHNOLOGIES Attacks on anonymization services [14, 33, 52, 68, 77, 82, 92, 101, 110, 163, 167], as well as the previous contribution, have shown that even slight alterations to an anonymity attacker model can cause a system to fail its purpose. Additional anonymity-enhancing technologies are then required to overcome the challenge of new attacks.

This thesis proposes two novel anonymity-enhancing technologies for anonymous Pub/Sub as the third contribution. These technologies have been published in [36]. The first technology is called *probabilistic forwarding* and adapts the related concepts of cover traffic and mimic traffic [57] to Pub/Sub. The concepts are adapted in such a way that they cannot be abused by known attacks and such that a combination of other technologies is possible.

The second technology is called the shell game (SG) and shuffles an overlay network—the middle layer in Figure 1. The overlay is shuffled in such a way that an attacker cannot de-anonymize participants by observing the overlay, e.g., using topological properties.

ANONYMOUS OVERLAY OPTIMIZATION Once an overlay network has been constructed in such a way that anonymity is preserved, it is subject to continuous changes and may degenerate in its properties [149]. Overlay paths may increase in length and thus cause message delay; forwarding load may become unevenly shared among the nodes and cause overload. Optimizing such an anonymous P2P is particularly challenging as knowledge about such an overlay is limited by design because of the anonymity requirement.

As a fourth contribution, this thesis introduces an approach for anonymous load balancing as part of the community management building block in Chapter 4. This approach establishes random groups of neighboring nodes in one or more overlay networks. A group then cycles a probabilistic data structure with each group member's load status. Overloaded group members can then reason about suitable handover candidates.

PUBLICATIONS This thesis has been published in 7 publications as listed on Page ix and summarized in this section [35–40, 59]. Most notably, the building blocks and anonymity-enhancing overlay technologies have been published by Elsevier as a highly visible article in the Computer Communications journal.

Jörg Daubert et al. „AnonPubSub: Anonymous publish-subscribe overlays.“ In: *Elsevier Computer Communications* 76 (2016), pp. 42–53. ISSN: 0140-3664. DOI: <http://dx.doi.org/10.1016/j.comcom.2015.11.004>. URL: <http://www.sciencedirect.com/science/article/pii/S0140366415004211>

Moreover, the publication on anonymity attacks [39] has been awarded by the Institute of Electrical and Electronics Engineers (IEEE) with a travel grant.

1.3 STRUCTURE OF THIS THESIS

The remainder of this thesis is structured as follows: Chapter 2 establishes the background in Pub/Sub followed by an introduction into anonymity. The introduction to anonymity comprises the foundational concepts, systems that shaped the domain of anonymity, and an overview of anonymity metrics. This introduction is followed by a discussion of related anonymization services, their advances, and lessons to be learned. The chapter then complements the anonymization services with an introduction to P2P systems and network simulation, a quantitative evaluation method for such systems.

Related anonymous Pub/Sub systems are analyzed in Chapter 3. For that, the chapter first introduces a formal model for Pub/Sub that serves as the foundation for an objective analysis. This model is then complemented by a detailed assessment of anonymity and corresponding sub-requirements in the domain of Pub/Sub. The requirements are succeeded by a discussion of attacker models. The chapter then introduces an attack based on the system and attacker model. To analyze related work in detail, the chapter proposes seven building blocks of anonymous Pub/Sub and introduces common technologies used for each building block, followed by a discussion of the related work.

Chapter 4 presents a novel construction for anonymous Pub/Sub. The chapter explains this construction according to the building blocks from the previous chapter, adapting existing technologies and proposing new approaches. This chapter also introduces the anonymity-

enhancing technologies, probabilistic forwarding, and the shell game, as well as the anonymous overlay optimization.

The system and various extensions of Chapter 4 are evaluated in Chapter 5. The evaluation first provides a qualitative discussion of the security requirements established in Chapter 2. An extensive quantitative evaluation via a network simulation supports the qualitative discussion and establishes parameter values for anonymity-enhancing technologies. A POC with an empirical assessment concludes the evaluation.

Chapter 6 summarizes this thesis and highlights the contributions and novel insights gained from the evaluation. This thesis then concludes with an outlook on future research directions.

BACKGROUND

This chapter introduces concepts as well as the necessary background for the domains of Pub/Sub and anonymous communication. These concepts will be used in the state-of-the-art discussion in Chapter 3, the contributions of this thesis in Chapter 4, as well as the evaluation (Chapter 5).

The chapter is structured as follows: Section 2.1 introduces the Pub/Sub paradigm. Section 2.2 introduces the concept of anonymity, the state-of-the-art in anonymization services (Section 2.3), and metrics. Section 2.4 introduces P2P systems, followed by a short overview of network simulation in Section 2.5.

2.1 PUBLISH/SUBSCRIBE

Pub/Sub is a content-based routing scheme [152]. Pub/Sub differs from other distributed programming patterns such as inter-process communication (IPC), remote procedure call (RPC), distributed objects, and tuple spaces. These differences are the loose coupling between sender and receiver, many-to-many communication, and information dissemination based on interest and filters [51].

Loose coupling allows exchanging messages between sender and receiver, whereas sender and receiver are not required to know each other. Such decoupling between sender and receiver is achieved via a proxy in between both. Loose coupling is advantageous to provide anonymity, as senders and receivers are not required to know each other—senders may remain anonymous towards receivers and vice versa. However, senders and receivers do not remain anonymous towards the proxy.

Many-to-many communication enables a sender to disseminate messages to multiple receivers at once. For that, the sender addresses the message to a group rather than a single recipient. Having an arbitrary number of receivers also benefits anonymity of receivers as they can blend with the group. Likewise, a receiver may obtain messages from many senders, too. Senders, therefore, blend with the group of senders and can be anonymous towards receivers.

The dissemination based upon interests and filters rather than recipient addresses is key for loose coupling. However, multiple interests of a receiver can be used to create a profile. Prior research [105] has shown that such profiles can be correlated with profiles of other data sources. Such a correlation may lead to a re-identification of the receiver, and thus to a violation of the receiver's anonymity.

Example 1. *Narayanan and Shmatikov used a pseudonymized dataset containing movie ratings and de-anonymized the raters [105]. In this case, all ratings provided by the same pseudonym constitute a profile. As similar pro-*

files can be found in public movie rating databases having an associated *ID* (e-mail address), these *ID*s can be associated with the pseudonyms as well. As a consequence, the pseudonymized participants become de-anonymized with their e-mail address. In their analysis, the authors have discovered that eight ratings per profile are already sufficient to de-anonymize 99% of the profiles within the pseudonymized dataset.

2.1.1 Concept and terminology

This section briefly introduces the core concepts of *Pub/Sub* and defines the terminology that will be used throughout this thesis. This section first provides a short overview of the *Pub/Sub* concept, then elaborates on comparison (or filtering) methods, followed by subscription methods and message dissemination.

OVERVIEW A basic *Pub/Sub* system as depicted in Figure 3 contains participants fitting one or more of the following roles: the *publisher*, an *event service*, and the *subscriber*. The publisher is the producer (sender) of *events*, and publishes those to the event service. The subscriber is the consumer (receiver) of events. To receive events from the event service, the subscriber first places a *subscription* at the event service. Upon receiving a publication from the publisher, the event service compares the publication with the subscription, and then sends a *notification* to the subscriber (push mechanism).

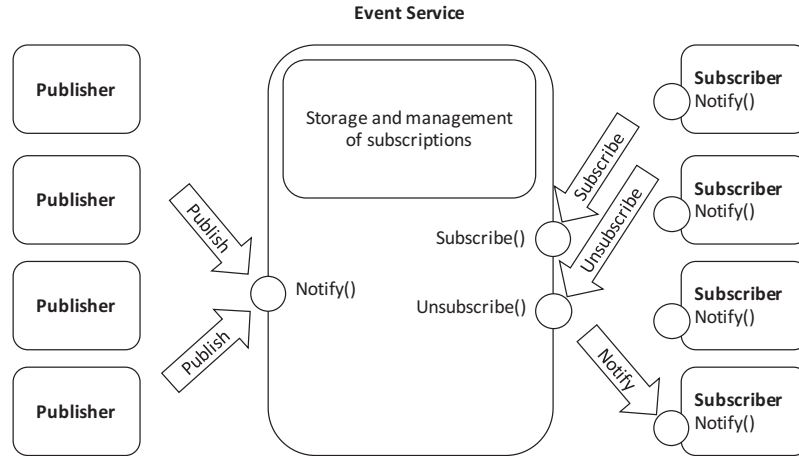


Figure 3: A simple object-based publish/subscribe system. [51]

COMPARISON METHODS The event service decides which notifications the subscriber receives from the publisher. For that, the event service compares notifications with the subscription. For that, several methods of comparison are known:

TOPICS With topics, notifications are annotated with one or more topics, e.g., keywords. A subscription is also expressed towards a topic. The event service, therefore, compares, if the notification contains the topic(s) specified by the subscription.

SUBJECTS With subjects, notifications are annotated with a subject. Compared to topics, subjects express additional context, comparable to Usenet groups and DNS. With such expressions following the reverse DNS notation, wildcards can be used in the subscription.

TYPE With types, subscriptions specify the type of notification. Such types can refer to the data type, e.g., text and binary, or object type.

CONTENT With content, subscriptions can specify filters over the full notification, including its content. For instance, the subscription can express predicates (key, operator, value) on key/value pairs. Filters over multiple notifications are possible as well.

Additional comparison methods exist: spatial comparison [70], semantic comparison [9, 20, 72, 118, 139, 146], parametric content-based comparison [78]. However, this methods primarily resemble special cases of the aforementioned methods.

Example 2. *The Java message service (JMS) [41] is a Pub/Sub Application Programming Interface (API) used for Java enterprise applications. A subscription in JMS is called a selector. It combines subject, type, and content-based comparison methods. For the content-based comparison, an expression language derived from SQL92 [73] can be used to filter a property map (key/value pairs).*

Privacy-preserving Pub/Sub solutions tackle the challenge of realizing comparison methods via cryptographic (confidentiality) means [23, 103, 154]. For that, content-based comparisons pose the most challenging method. As this thesis is dedicated to anonymous rather than confidential Pub/Sub, the topic-based comparison will be used throughout the remainder of the thesis. Topic-based comparison introduces the least cryptographic complexity, and thus least deviates from anonymization mechanisms.

SUBSCRIPTION METHODS Pub/Sub with subscriptions causes processing overhead as the event service has to compare every notification with every subscription. If a publisher uses the same topic/subject/type for consecutive notifications, a comparison may not be necessary for consecutive notification. To overcome this drawback of subscription-based Pub/Sub, a variation with *advertisements* can be used.

With advertisements, a publisher first advertises the topic/subject/type it will use for notification to the event service. Together with the subscription, the event service establishes a fixed routing path between publisher and subscribers. As a consequence, no consecutive comparisons are necessary.

This thesis focusses on advertisement-based subscription methods. As topic-based comparison has been selected, notification distribution paths can be established via advertisements, as no additional comparison per notification is required.

MESSAGE DISSEMINATION A variety of methods are used to transport notifications from the event service to subscribers as shown in Figure 4. The node type *f* represents forwarders (brokers, event service), *R* transport layer multicast (TLM) capable routers, and *s* subscribers. The infrastructure nodes *f* and *R* are marked in gray, the subscribers *s* in white.

POINT-TO-POINT The event service establishes a direct connection to each matching subscriber and delivers the notification.

TLM The event service makes use of network- and transport layer multicast mechanisms [42], e.g., protocol independent multicast (PIM) [2, 55], to deliver a notification to multiple subscribers at once.

APPLICATION LAYER MULTICAST The event service uses a P2P multicast [8] to distribute the notification via overlay members to multiple subscribers at once.

BROKER NETWORK The event service distributes the notification to multiple brokers via application layer multicast (ALM). The brokers then disseminate the notification via PtP, TLM, or ALM to subscribers. The SIENA Pub/Sub system [19] uses this approach.

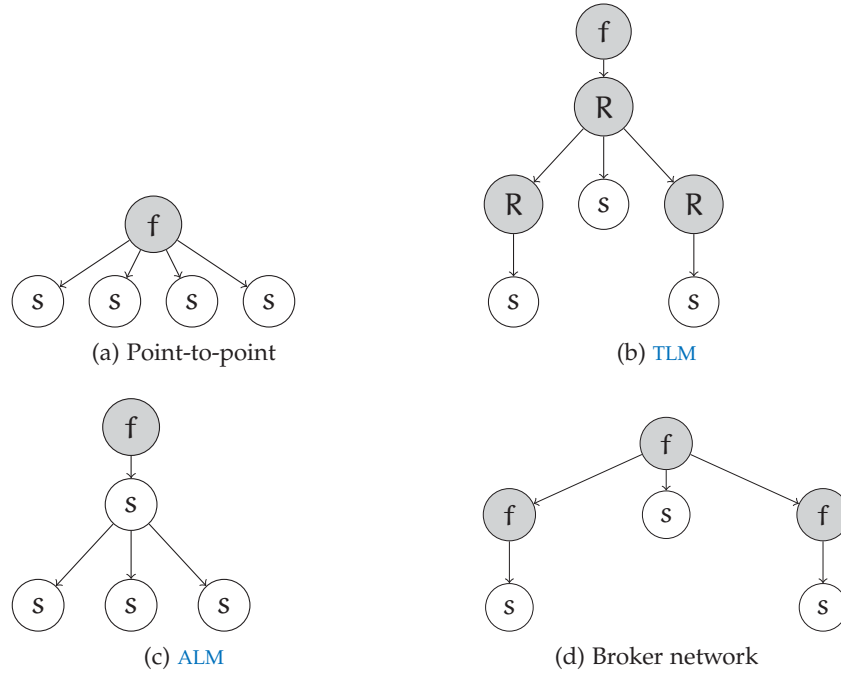


Figure 4: Methods for disseminating notifications

PtP connections are easy to implement and resilient. However, these connections only scale linearly with the numbers of subscribers. Foremost, point-to-point connections cannot provide publisher and subscriber anonymity towards the event service; the event service is required to know publisher and subscriber IDs to establish connections.

TLM mechanisms such as **PIM-SM** establish distribution trees and thus scale with the number of subscribers. Furthermore, **TLM** does not require the event service to know subscribers. However, **TLM** infrastructure entities, such as routers, learn the **IDs** of subscribers. Thus, subscriber anonymity against the infrastructure cannot be guaranteed.

ALM mechanisms also establish distribution trees and meshes, and thus scale with the numbers of subscribers as well. However, **ALM** relies on peers to execute the notification dissemination algorithm at the application layer. Thus, existing infrastructure such as routers can hardly be used for **ALM**, and new peers are required. According to Esposito and Ciampi [49], **ALM** appears to be the dominant solution for secure **Pub/Sub** system from academia.

In summary, **ALM** offers scalability as well as the flexibility to design the application-layer routing algorithms in an anonymity-preserving manner. The remainder of this thesis, therefore, focuses on **ALM**-based **Pub/Sub** approaches.

2.1.2 Requirements

A **Pub/Sub** system must comply with a plethora of functional and security requirements. Some of these requirements diametrically oppose anonymity. For instance, sender authenticity obviously conflicts with sender anonymity. For that reason, functional as well as security requirements have to be discussed in combination with anonymity.

Surveys [49, 51, 162] enumerate these requirements as follows:

CONFIDENTIALITY Confidentiality as the first requirements of the **CIA** triage enforces information not to be disclosed to an unauthorized participant. Such information occurs in three variations in **Pub/Sub**:

INFORMATION CONFIDENTIALITY No information must be disclosed to the event service, i.e., the infrastructure. This is particularly challenging as the event service requires information to make routing decisions, e.g., with content-based **Pub/Sub**.

SUBSCRIPTION CONFIDENTIALITY No information about a subscription—and thus about the interests of a subscriber—must be leaked to another subscriber and publisher.

PUBLICATION CONFIDENTIALITY No information about a publication (notification)—and thus information about a publisher—must be leaked to another publisher and unauthorized subscriber.

INTEGRITY Integrity as the second confidentiality integrity availability (**CIA**) triage requirement ensures the completeness and correctness of data. For **Pub/Sub**, Wang et al. [162] distinguish the following three variations:

INFORMATION INTEGRITY Information integrity requires that no information, e.g., stored data or message in transmission, is altered or lost without authorization.

SUBSCRIPTION INTEGRITY Subscription integrity requires that no subscription be altered or lost.

SERVICE INTEGRITY Service integrity requires that the event service is not altered. Counter examples are the failure of a broker and the compromise of a broker. Service integrity in combination with information and subscription integrity ensures that that no subscription is lost, and that all matching notifications are delivered without unauthorized modifications. This requirement is also called reliability [50]

AVAILABILITY Availability as the third CIA triage requirement ensures that information is available when needed. That includes that notifications are delivered without delay. Therefore, the event service must be available when needed. Counterexamples are failures of the event service due to denial of service (DoS) attacks. While a distributed denial of service (DDoS) might be initiated intentionally by an attacker, subscriptions and publication may also cause overhead leading to a DoS. Examples of such subscriptions and publications are subscriptions that are computationally expensive to evaluate as well as wildcard subscriptions that cause the mass delivery of notifications, and publications that are large. This requirement can be also called scalability [50].

AUTHENTICATION Authentication in Pub/Sub requires receivers to be able to verify the identity of senders and vice versa. That applies to PtP communication, e.g., publishers and event service, and to end-to-end (E2E) communication, e.g., subscriber and publisher.

ACCOUNTABILITY Accountability requires all actions, e.g., sending and receiving a message, to be associated with an actor. Furthermore, the actor cannot be able to dispute the action. For instance, it must be clear if a subscriber received a notification, and if so, that the subscriber cannot deny the reception.

USER ANONYMITY User anonymity requires users—subscribers and publishers—not to be identified and observed by other users or the event service. That is, users maintain anonymity in the sense that they cannot be identified, e.g., via an IP address. Furthermore, the actions of a user cannot be linked together, e.g., profiling is not possible. This requirement directly conflicts with authenticity (the identity of the user must be verifiable) and accountability (every action is linked to an actor). An anonymous Pub/Sub system should therefore not ignore these requirements but rather balance them, e.g., with authenticity.

LOW OVERHEAD This thesis extends the list of requirements [50, 162] by minimal overhead. First, minimal overhead serves as sub-goal, e.g., for availability/scalability. Second, mechanisms to protect anonymity introduce additional overhead. For instance, despite the computational overhead caused by cryptography, anonymization services hardly reach the predicted performance [145].

LOW COMPUTATIONAL OVERHEAD Low computational overhead requires that computation, in particular for the Pub/Sub functions *match* and *cover* (down below).

LOW SPACE OVERHEAD Low space overhead requires that the consumed space regarding memory be as low as possible. Low space overhead can be also considered a sub-requirements of availability/scalability.

LOW SIGNALING OVERHEAD Low signaling overhead requires the number and sizes of messages to be as low as possible. Signaling overhead occurs in Pub/Sub regarding advertisements, subscriptions, and notifications. Further messages may occur, e.g., confirmation messages to ensure integrity. Mechanisms to protect anonymity may introduce signaling overhead by mixing messages [21].

To comply with service integrity, an event service needs to *match* subscriptions with publications. Optionally, the event service may aggregate subscriptions by checking if one subscription is already *covered* by another. This function addresses scalability.

MATCHING Assuming a set of attributes A , given a notification m_{notif} as well as a subscription m_{sub} , and a function $\mathcal{A}(m)$ that extracts all attributes from a message m , $match(m_{notif}, m_{sub})$ returns *true* if $\mathcal{A}(m_{sub}) \subseteq \mathcal{A}(m_{notif})$, i.e., the notification matches all attributes of the subscription.

COVERING Given two subscriptions, $m_{sub,1}$ and $m_{sub,2}$, $cover(m_{sub,1}, m_{sub,2})$ returns *true* if $\mathcal{A}(m_{sub,1}) \subseteq \mathcal{A}(m_{sub,2})$, i.e., $m_{sub,1}$ is more general and covers $m_{sub,2}$.

The match operation is mandatory and must be computable by the event service, i.e., every broker. However, this operation also reveals information that may violate confidentiality and user anonymity. Implications regarding anonymity are deeply anchored within Pub/Sub. The next section elaborates further on anonymity.

2.2 ANONYMITY

The terms anonymity and privacy are intrinsically related. Privacy refers to the ability to seclude oneself. Privacy can be understood as a fundamental right, as Maslow [99] enumerates the “need for privacy” as a core property of self-actualization. Poore specifies [116] the subject, i.e., the *what*, of privacy as “[...] information leaving

the control of the person whose information it is [...]”. Such information indicating the identity of a person is called personally identifiable information (PII) [106]. Further definitions address the protection of PII and thus privacy, e.g., Lessig [90] mentions “empowerment to control” as one requirement for privacy. Standards such as ISO/IEC [74] and Common Criteria [29] decompose the requirement class privacy into anonymity, pseudonymity, unlinkability, and unobservability. Anonymity is defined as “Anonymity requires that other users or subjects are unable to determine the identity of a user bound to a subject or operation”. Following these definitions, this thesis considers anonymity as a part of privacy. In particular, anonymity as requirement concerned with the protection of PII.

A technical solution to anonymous communication was first proposed by Davin Chaum in 1988 via the *Dining Cryptographers Problem* [22]. Since then, many solutions for anonymous communication have emerged. Some of them leverage Chaum’s solution to the dining cryptographers problem. Dining cryptographers networks (DC-nets) rely on proxy servers or use group communication. The following two subsections introduce the concepts of anonymous communication, as well as metrics to measure the success of these concepts.

2.2.1 Concepts

With an anonymization service, participants usually use a public communication channel, e.g., the Internet, and are thus exposed to various attackers [48]. Anonymization services attempt to fulfill the following requirements in such a scenario [113]:

UNLINKABILITY Two or more messages cannot be linked together, i.e., to the same sender, to the same receiver, or to the same attribute

SENDER ANONYMITY No message must be linked to its sender.

RECEIVER ANONYMITY No message must be linked to any of its receivers.

RELATIONSHIP ANONYMITY No receiver must be linked to the sender and vice versa.

In the case of a continuous conversation between sender and receiver, the sender is also called *initiator*, and the receiver is called *responder*.

During the past decades, three main concepts to achieve these anonymity goals have emerged. First, group communication that ensures that senders and receivers “hide” within a group of potential sender and receivers. Second, proxy nodes that act “on behalf” of a sender or receiver. As a special version of a proxy, a MIX may provide better anonymity, but also causes latency. Third, DC-nets as a special case of group communication. The following paragraphs introduce each of the concepts.

GROUP COMMUNICATION Group communication follows the concept of hiding in a group, e.g., when a shared medium is used. For instance, a message addressed to many receivers at once provide receiver anonymity. While group communication can protect anonymity, additional protection such as encryption is required to ensure confidentiality.

Examples of this method are IP broadcast (receiver anonymity), IP multicast [PIM](#) [2]. Broadcast and multicast are however not suited for Internet-scale communication. On the one hand, IP broadcasts are limited to a subnet. Thus, only a few subscribers can be reached, which limits the user anonymity for receivers as well as scalability. On the other hand, [PIM](#) is typically restricted to subnets as well, but not due to technical but rather organizational reasons.

PROXY SERVERS Proxy servers relay messages on behalf of the original sender. Assuming that only the relayed messages are observable, proxy servers can provide sender anonymity. Multiple proxy servers can be connected to form a chain, a so-called *jondo*. Assuming that the last relayed message is not observable, such a proxy chain can provide receiver anonymity. First such systems have been implemented in the form of remailers for e-mail [111].

Proxy servers may also recode messages before relaying them. Such recoding can be performed by (re-) encrypting the message. The recoding ensures that an incoming message cannot be linked to an outgoing message. Hence, recoding attempts to protect unlinkability. However, if the proxy server relays messages in the same order as it received the messages, the outgoing and incoming messages can be still linked together according to their sequence, despite recoding. To overcome this issue, proxy servers reorder messages. For that, proxy servers collect several messages—batch processing—and relay them in shuffled order.

A proxy server, which recodes, shuffles, and relays messages, is also called a *MIX* [21]. A collection of MIX nodes forms a [MIX-net](#). [MIX-nets](#) may occur in the topology of *free-routes* and *cascades*. With *free-routes*, every sender chooses a sequence of MIX nodes independently. With *cascades*, the sequence of MIX nodes is predetermined, only the first MIX node can be chosen by the sender. *Free-routes* are considered the preferable variation [45], as they are resilient against MIX node failures.

Figure 5 depicts the types of [MIX-nets](#). From left to right: a cascade with 2x2 nodes, a stratified MIX with 2x2 nodes—messages can be relayed from any node to any node, and a 4x2 free-route MIX.

A MIX is also called a *high-latency* anonymization service, whereas a proxy is called a *low-latency* anonymization service. The high latency of a MIX occurs as a MIX shuffles messages, i.e., it collects several messages rather than forwarding those messages directly. Depending on the number of messages a MIX collects—the batch size—the latency of a MIX may increase.

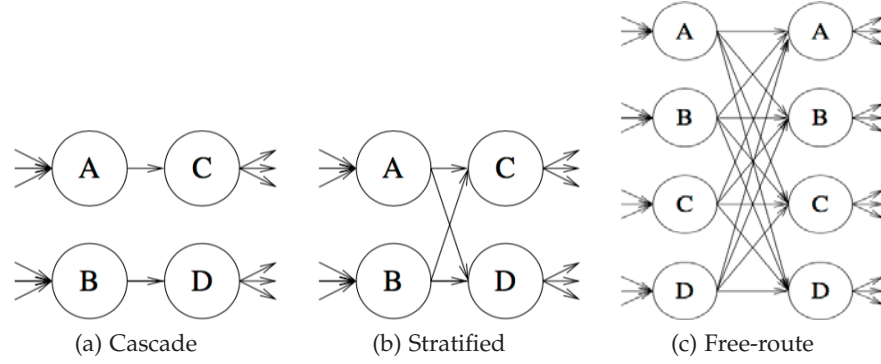


Figure 5: Types of MIX-nets [45]

DC-NETS dining cryptographers networks (**DC-net**) are a principle of secretly sharing a message within a group introduced by Chaum in 1988 [22]. With **DC-nets**, noise is added to every message, so that neither a global observer nor any participant learns the contents of the messages. After all messages are exchanged, the noise balances itself out, and the content of the message is revealed, but the sender remains anonymous.

With the original and most basic version of a **DC-net**, three cryptographers desire to determine if any of them paid the bill for the dinner. I.e., at most one cryptographer would send the message 1 (paid the bill); all other would send the message 0 (did not pay the bill). The cryptographers establish pairwise shared secrets via a coin flip—0 or 1 depending if head or tail. The cryptographers then “encrypt” their message by adding the shared secret via the logical xor \oplus to the message, and send the encrypted message to the corresponding neighbor. Every cryptographer can establish if someone sent 1 by combining the two received messages with their message via \oplus again. As the same shared key—the noise—is used twice, it balances itself out by the definition of \oplus .

2.2.2 Anonymization systems

The introduced concepts for anonymous communication have been implemented in a plethora of systems. This Section introduces selected systems that have a notable impact or have introduced novel mechanisms to anonymous communication. The state-of-the-art in anonymous **Pub/Sub** systems will be discussed in Chapter 3.

ONION ROUTING (TOR) the onion router (**Tor**), first proposed by Syverson [151] and then revised by Dingledine [44], implements a low-latency MIX network. Compared to a free-route MIX, **Tor** omits the shuffling step and forwards messages immediately (low-latency). However, **Tor** uses the concept of onion routing to encrypt a message in such a way that every node of a free-route path only learns its immediate predecessor and successor.

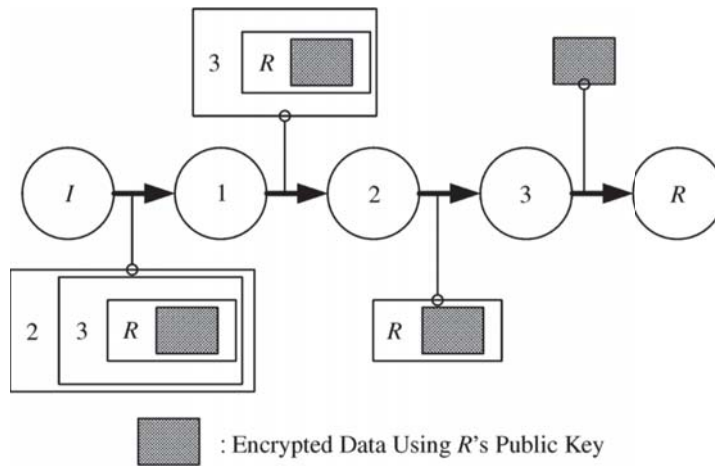


Figure 6: Illustration of onion routing [168].

With onion routing [151] the initiator of a message encrypts the message multiple times and thus creates layers of encryption around the message—the onion as shown in Figure 6. An onion layer is formed as follows: the initiator *I* takes the message, appends the address of a node *R*, and encrypts the message with the public key of a node 3. As a result, node 3 can decrypt the message (peel off an onion layer), read the address of node *R*, and forward the message to node *R*. Following this approach, multiple layers of encryption can be applied to a single message, starting with the responder, the last Tor proxy node—also called *exit guard*, the second to last Tor proxy node, [...], and the first Tor proxy node—called *entry guard*. As every Tor proxy node possesses its own secret/public key pair, every node can only peel off one onion layer, read the address of the next node, and get to know the previous node as the sender of the message.

FREENET (DARKNET MODE) Freenet is a P2P information storage, search, and retrieval system [24]. Freenet contains two operation modes, *opennet* and *darknet*. The *opennet* mode of Freenet already addresses anonymity by omitting E2E addresses in information search. Freenet uses flooding (recursive routing) instead [160]. That means that neighbors act as proxies and forward requests on behalf of their neighbors. The *darknet* mode further restricts the *opennet* mode by limiting neighbors to trusted nodes. With that, Freenet attempts to protect anonymity in the presence of malicious insiders. As users only connect to neighbors they trust, they can exclude malicious insiders in their neighborhood. As their trusted neighbors also only connect to trusted nodes, malicious insiders within the extended neighborhood are unlikely. As the *darknet* mode of Freenet uses recursive routing, no node IDs, e.g., IP address, leaves the direct, trusted neighborhood. This type of overlay is therefore also referred to as membership-concealing overlay network (MCON) [160]. Sender anonymity, receiver anonymity, unlinkability and relationship anonymity are assumed to be fulfilled in a *darknet*.

Darknets have emerged from a mode in Freenet to a generic concept in anonymous communication, e.g., darknets with Tor [80], where

proxies are selected based upon their trustworthiness. Darknets are not to be confused with the *dark web* [7], the non-indexed part of the Internet.

2.2.3 Metrics

Anonymization concepts and system are intrinsically tied to the question, how well they fulfill anonymity goals. In the attempt to answer that question, several metrics have been proposed to measure anonymity requirements. This thesis summarizes the dominant metrics set size, probability, entropy, and Euclidean distance.

SET SIZES Anonymity can be measured by the size of an anonymity set.

Definition 1. “Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.” [112, 113]

Following this definition, sender anonymity can be measured by the number of all potential senders (Figure 7 left). Likewise, the receiver anonymity can be measured by the number of all potential receivers (Figure 7 left). A similar definition of anonymity is used by k-anonymity [150], DC-nets [22] and MIX-nets [21].

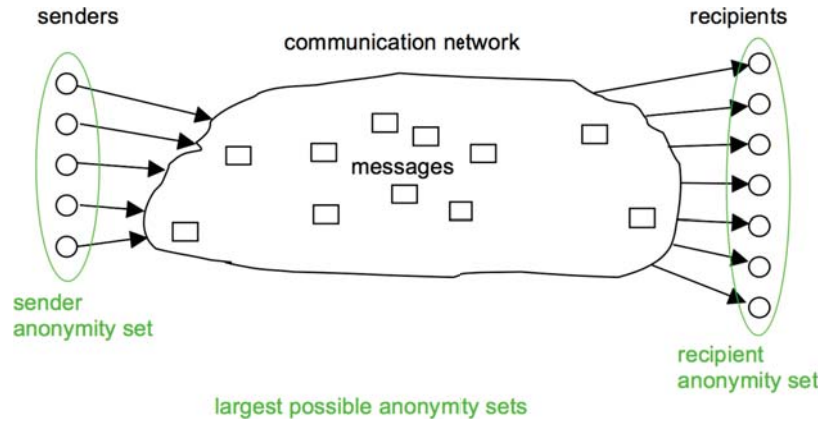


Figure 7: Anonymity set sizes. [112]

k-ANONYMITY The case of k-anonymity well illustrates the challenge of defining anonymity metrics. The original approach published by Sweeney et al. in 2002 defined a record in a medical dataset as k-anonymous (with $k \geq 2$) if for every key (equivalence class) of the record, at least $(k - 1)$ other records in the dataset shared the same value (are indistinguishable). However, k-anonymity could be established by just inserting $(k - 1)$ copies of every record into the dataset. An anonymity attacker with knowledge of this approach could trivially remove these copies.

To overcome the drawbacks of the k-anonymity approach, Machanavajjhala et al. proposed l-diversity in 2007 [98]. They enforce that every key in the dataset should, at least, occur with l diverse values (every equivalence class has, at least, l well-represented values).

With that property, the combined anonymity metric becomes the tuple (k, l) . While inserting copies of the same record would still increase k , l would not improve.

With l -diversity, even if the values of an equivalence class are l well represented, the sensitivity of the values may differ. E.g., having been tested positive for a terminal illness is more sensitive than having been tested negative. To overcome this issue, Li et al. proposed t -closeness in 2007 [91]. With t -closeness, values of an equivalence class have to be close to each other with a maximum distance of t . As a result, the distribution of values of an equivalence class show less skewing and are less sensitive. The anonymity metric becomes the triple (k, l, t) .

With t -closeness, it is still assumed that the dataset is only anonymized and released at once. However, some dataset may be released in intervals with incremental updates. For instance, medical records of patients may be released quarterly. In such a case, some records may persist between multiple releases or not. An anonymity attacker may correlate information over multiple “snapshots” (data release) to reduce the anonymity of records. To overcome this issue, Xiao et al. proposed m -invariance in 2007 [169]. With m -invariance, at least, m records must persist (remain invariant) across two consecutive dataset releases. As a result, the anonymity metric becomes the quadruple (k, l, t, m) .

The excursion on the well-studied subject of k -anonymity shows that it is difficult to establish an anonymity metric. In particular, it seems to be difficult to show that a metric cannot be attacked. For instance, Kesdogan et al. showed [83] that anonymity sets can be reduced significantly depending the attacker’s capabilities.

ENTROPY Set sizes as anonymity metric do not model additional knowledge of an anonymity attacker. For instance, regarding sender anonymity, an attacker could consider some potential senders more likely than others. Set sizes cannot incorporate this knowledge. To overcome this drawback, Serjantov and Danezis propose *entropy* [132] as an anonymity metric, based on the Shannon entropy [133].

The entropy is defined as a scalar value over a probability distribution of all potential subjects Ψ . For every subject $u \in \Psi$, p_u (Equation (2)) models the a posteriori probability—after the attacker observed messages—of u acting in role $r \in \mathcal{R}$. \mathcal{R} can be either sender or receiver. The Shannon entropy \mathcal{S} (Equation (1)) provides the entropy over this probability distribution p [132].

$$\mathcal{S} = - \sum_{u \in \Psi} p_u \log_2(p_u) \quad (1)$$

$$p_u = \mathcal{U}(u, r) \quad (2)$$

Entropy as anonymity metric allows expressing fine-grained knowledge p_u about every subject u . An entropy of $\mathcal{S} = 0$ indicates no anonymity. An entropy of $\mathcal{S} = \log_2 |\Psi|$ indicates perfect anonymity, i.e., the anonymity set has the size $|\Psi|$.

Diaz et al. extend this notion by the so-called *degree of anonymity* [43], a value derived from the normalized entropy that expresses how much the attacker learned by observing messages. Following the notation of Serjantov and Danezis, the degree of anonymity $d \in [0, 1]$ is defined as (Equation (4)), the relation of the a-posteriori entropy \mathcal{S} about the maximum entropy H_M (Equation (3))—the perfect anonymity.

$$H_M = \log_2 |\Psi| \quad (3)$$

$$d = 1 - \frac{H_M - \mathcal{S}}{H_M} = \frac{\mathcal{S}}{H_M} \quad (4)$$

Therefore, the degree of anonymity represents a normalized scalar metric, which is independent of the population of subjects. With $d = 0$, the system provides no anonymity; with $d = 1$, the system provides perfect anonymity.

Clauss and Schiffner argue that Rényi entropy [122] can be used as well as an anonymity metric [25]. Following Equation (5), the Rényi entropy $H_\alpha(P)$ is a generalization of the Shannon entropy via the parameter α . With $\lim \alpha \rightarrow 1$, the Rényi entropy converges to Shannon entropy. Clauss and Schiffner furthermore point out that entropy metrics are heavily influenced by outliers.

$$H_\alpha(P) = \frac{1}{1 - \alpha} \log_2 \sum_{u \in \Psi} p_u^\alpha \quad (5)$$

Anderson and Lundin propose a scaled version of Shannon entropy as anonymity metric [5]. The scaled anonymity set size A (Equation (6)) is directly derived from the Shannon entropy (Equation (1)). Values of $A = |\Psi|$ denotes the maximum anonymity set size; $A = 1$ denotes no anonymity, i.e., a uniquely identified sender (or receiver respectively).

$$A = 2^{\mathcal{S}} \quad (6)$$

Tóth et al. argue that a so-called *global metric*, such as entropy, is insufficient as an anonymity attacker could already exploit a small deviation of p_u from the a priori probability. Therefore, they propose the local metrics *source-hiding property* Θ and *destination-hiding property* Ω . An anonymization service is considered source-hiding with parameter Θ if Equation (7) holds. Likewise, an anonymization service is considered destination-hiding with parameter Ω if Equation (8) holds. Also, statements about global metrics (Shannon entropy \mathcal{S}) can be derived from the local source-hiding property via Equation (9).

$$\forall p_u \in \Psi : p_u \leq \Theta \quad (7)$$

$$\forall p_u \in \Psi : p_u \leq \Omega \quad (8)$$

$$\mathcal{S} \geq -\log_2 \Theta \quad (9)$$

Reiter and Rubin use six degrees of anonymity [120]: absolute privacy, beyond suspicion, probable innocence, possible innocence, exposed, and provably exposed. This metric is local, i.e., it is unique for every subject, and it incorporates the graph topology of the crowd as well as assumptions regarding the anonymity attacker. For instance, Equation (10) describes the condition of *probable innocence* for the sender. Here, $p_f > 0.5$ describes the probability for forwarding in the system; c denotes the number of colluding anonymity attackers.

$$|\Psi| \geq \frac{p_f}{p_f - 0.5}(c + 1) \quad (10)$$

SUMMARY In summary, no perfect metric to measure anonymity seems to exist. While the Crowds metric is intuitive, it is also highly specific to the Crowds system. The set size metric is also easy to comprehend but lacks the capability to model fine-grained attacker knowledge. The entropy-based metrics seem to be the most comprehensive metrics so far.

2.3 ANONYMIZATION IMPLEMENTATIONS & SERVICES

Based on the concepts and systems summarized in Sections 2.2.1 - 2.2.2, several ready to use implementations and even commercial services have emerged. While these implementations and services hardly introduce relevant novel concepts, they have been studied extensively regarding anonymity attacks. Therefore, this Section briefly discusses the most cited implementations & services, as well as respective attacks and lessons to be learned.

2.3.1 *anon.penet.fi*

The service *anon.penet.fi* was a centralized proxy service for e-mail. The proxy receives messages from a sender, assigns a pseudonym to the sender, stores the sender \leftrightarrow pseudonym mapping, replaces the sender in the messages with the pseudonym, and forwards the message to the receiver. Due to the pseudonym mapping, the proxy can assign responses to a pseudonym to the correct originator.

While the service was successful and handled many messages per day, lawful copyright claims forced the operator to disclose pseudonym mappings [48]. This case shows that a centralized pseudonym mapping causes issues safeguarding these mappings.

2.3.2 *Cypherpunk remailers*

This service extends the concept of *anon.penet.fi* with multiple proxy servers. For that the onion routing concept is used, i.e., the senders adds an encrypted header for every proxy to the e-mail. To enable the receiver to respond to an e-mail, the sender can also add a reply

block to the message. This block contains the headers describing the proxy chain to be used by the receiver to respond [48].

Cyberpunk remailers share concepts with today's onion routing. However, these remailers do not pad messages, and may not implement good shuffling of messages (high batch size). Nevertheless, e-mail typically allows for some latency, and thus would allow remailers to implement good shuffling.

2.3.3 *Mixminion*

Mixminion [32] is another incarnation of the remailer concept. Compared to Cyberpunk remailers, Mixminion nodes maintain hashes of relayed messages for some time to prevent replay attacks. Likewise, Mixminion also restricts the reply blocks to be used only once. Mixminion also introduces the concept of directory servers to discover Mixminion nodes. This concept is still used today by Tor.

2.3.4 *Crowds*

Crowds [120] applies a routing scheme via proxies—not unlike hot potato routing—to anonymize senders. Within crowds, every node runs a routing process called jondo. Therefore, Crowds is a P2P approach. Every jondo belongs to a crowd of jondos, thus the name of the system. Jondos are assigned to crowds via the blender process. Such a decision can be made for instance based on physical locality.

Whenever a sender wants to reach a receiver, it first establishes a path within the crowd. Compared to Tor (cf. next paragraph) circuits, the sender does not specify this path in Crowds. Every jondo, including the sender, just picks the next hop. For that, every jondo that receives a message chooses with a forwarding probability $p_f > 0.5$ to forward the message to another jondo, or to the receiver otherwise. Once this path is established, sender and receiver communicate via this path.

Crowds protects against de-anonymization from malicious insider threats. No jondo can tell, if the previous jondo is just a relaying jondo or the original server. However, Crowds does not apply any mixing or padding technology. Therefore, an anonymity attacker who observes the message flow can trace a path in Crowds back to the origin. Extensions of Crowds exist, for instance, to relay messages in between crowds [131].

2.3.5 *Tarzan*

Tarzan [57] constructs circuits consisting of proxy servers to relay UDP packets. Like Crowds, all nodes act as proxies. As there is no “crowd” assigned to Tarzan nodes by a central server, Tarzan relies on gossiping to discover neighbors. With gossiping (P2P), Tarzan connects to a known neighboring node, and asks it for more neigh-

bors [48]. Using this approach, Tarzan nodes learn about proxies that can be used to build circuits.

To prevent anonymity attackers from tracing message flows, Tarzan uses cover traffic between nodes. For that, every node selects a couple of Tarzan nodes with which it exchanges cover traffic (mimic traffic). These nodes are called *mimics*. The mimic selection process is based on a deterministic function and publicly verifiable information such as IP address and date [48]. Having this information, mimics can verify their selection, and thus prevent malicious insiders from forcing mimics to generate cover traffic and overload the system.

2.3.6 Tor

Tor [44] is the reference implementation of onion routing as introduced in Section 2.2.2. This section elaborates on additional functions of Tor beyond onion routing, as well as attacks against Tor.

2.3.6.1 Hidden services

The concept of onion routing is supposed to protect sender anonymity, but not receiver anonymity. An initiator is required to know the address of the responder. The second generation of Tor introduced a concept to protect receiver anonymity as well. To reach an unknown responder, Tor uses the notion of hidden services. With hidden services, the initiator addresses a particular type of service but does not care who provides this service. Hence, the name hidden services.

With hidden services, the service provider (responder) establishes a Tor circuit but does not contact any node outside of the Tor network. The service provider rather instructs the exit guard of the circuit to keep the TCP port of the circuit open. As a result, every initiator contacting the guard using the specific port will reach the service provider, but will not reveal its identity. To inform initiators how to reach the guard, the service provider publishes a hidden service listing, containing guard, TCP port, public keys, in a central repository.

Assuming that the initiator also establishes a circuit before connecting to the service provider's circuit, both, initiator and responder, each control a circuit to ensure their anonymity. Tor recommends changing circuits on a regular basis to prevent prolonged traffic correlation by a malicious onion router. Hidden services, however, require the service provider to publish every new circuit (the guard) in the central repository. Furthermore, initiators have to obtain this information. As a result, regular circuit changes are less practical with hidden services.

2.3.6.2 Attacks

Tor [44] is one of the most popular anonymization services today. Consequently, Tor has been subject to attacks and in-depth analysis. These attacks provide valuable insights regarding the construction of anonymization services and anonymous Pub/Sub systems, too. Tor at-

tacks can be categorized in passive and active attacks. With passive attacks, an attacker observes network traffic. This attacker can be a malicious onion router that is part of a [Tor](#) circuit, or an attacker that can perform large scale Internet traffic analysis. With active attacks, an attacker influences the behavior of the [Tor](#) system deliberately. Such an attacker can take the role of an initiator, which establishes a [Tor](#) circuit, and generates traffic, e.g., to overload the circuit. An active attacker can also take the role of an onion router that delays messages or marks packets.

CONGESTION With the congestion attack, an attacker attempts to exhaust resources of onion routers to force initiators to pick other onion routers for their circuit [52]. This attack can be used to concentrate traffic onto onion routers controlled by the attacker. This attack is an active attack and can be used in conjunction with other active or passive attack to de-anonymize [Tor](#) users.

PACKET SPINNING The packet spinning attack [110] is a more elaborate version of the congestion attack. It abuses the fact that [Tor](#) nodes cannot detect loops in circuits, and thus create such loops and cause packets to spin in this loop forever. As a result, this attack causes a congestion of the involved onion routers as well, also causing initiators to select other onion routers. Both congestion attacks work as they cause signaling overhead as well as computation overhead—onion routers have to perform cryptographic operations when relaying messages after all. Whenever an initiator attempts to establish a circuit, it contacts every onion router which is supposed to be part of the circuit one after another via the growing [Tor](#) circuit. If an onion router does not respond within a timeout, the user will choose another onion router.

TRAFFIC ANALYSIS [82] The traffic analysis attack attempts to correlate incoming and outgoing traffic of onion routers. Ideally, every conversation between initiator and responder follows a unique traffic template (a function mapping from time to bandwidth). If an attacker can trace this pattern through all onion routers of a circuit, the attacker can link initiator and responder. While this attack would require global observing capabilities of the attacker, the attacker can be refined to work with a weaker attacker model [101]. As everyone can establish arbitrary [Tor](#) circuits, a single active attacker can establish circuits and measure the delay (and thus load) of individual onion routers. Together with a malicious responder that generates load patterns, the attacker can predict which other onion routers are in the circuit based on the observed delay pattern.

SNIPER The sniper attack [77] is a special version of the congestion attack. It uses a weakness of [Tor](#)'s flow control to exhaust the memory of an onion router quickly with minimal resource consumption. This attack is possible as [Tor](#) does not use [E2E](#) dupli-

cate detection. Therefore, it is possible to instruct onion routers to send large message many times with a small request.

CELL COUNTER The cell counter attack [92] is one example of marking traffic in a Tor circuit. Marking traffic helps colluding attackers to trace message flows. In an ideal case, the attacker controls the entry guard—the first onion router of a circuit—and the last onion router of a circuit (exit guard). Then, the entry guard would mark packets in a way such that the exist guard could detect these markings. As a result, the attacker could link initiator and receiver. The cell counter attack is one specific method to mark packets. As Tor does not use TCP streams to communicate along the circuit, and as the payload is encrypted, only metadata fields of Tor can be used for marking. The cell counter is a metadata field used by Tor for flow control, similar to the TCP sequence number. Slight alterations to the cell counter can be used to mark circuits.

BAD APPLE The bad apple attack [14] uses an onion router to perform man in the middle (MITM) attacks (active attacks). The attack exploits that communication between the Tor exit guard and the receiver may occur without confidentiality and authenticity protection. A malicious exit guard can, therefore, read the message from the sender and modify the response. In the example [14], the exit guard modified the response from BitTorrent tracker to introduce malicious peers. Assuming that the targeted node will directly connect to the malicious peer without Tor, the target node can be de-anonymized.

FINGERPRINTING Fingerprinting attacks [68, 163] exploit the fact that the application layer may leak information that is reflected in Tor circuits by a fingerprint. With a database of fingerprints, every onion router of a circuit can correlate this circuit with application layer receivers. For instance, with HTTP / a web page as application layer responder, the attacker can build a database of fingerprints from common websites. The fingerprint is then constituted by the timed sequence of requests (HTML, JS, CSS, images)—also called timing attack—and the sizes of the responses. With this information, a Tor entry guard can link initiators with web pages, i.e., link initiators and receivers.

STATISTICAL DISCLOSURE The statistical disclosure attack [33] assumes that some properties of anonymous communication remain invariant over time. Therefore, statistics can be used to reveal these invariant properties. For instance, if the communication between initiator and responder is invariant and the path through the anonymization service variant, simply counting messages of initiators and receivers would statistically reveal this pair. However, this attack requires an attacker that has global observation capabilities.

PREDECESSOR The predecessor attack [167] is a specific version of the statistical disclosure that has been applied to Tor. This attack assumes that it is possible to associate a Tor circuit with a responder, e.g., via the fingerprinting attack. Then the initiator would connect more often to a malicious onion router over time than any other onion router. Thus, via the statistics, the attacker could assume that this node is indeed the initiator and not an onion router. This attack assumes that the initiator creates many circuits over time to connect to the same responder. However, the current specification of Tor prevents the creation of a new circuit as long as the conversation between initiator and responder is ongoing.

2.3.7 *MorphMix*

MorphMix [121] is a MIXnet that does not rely on fixed MIX cascades or directory servers. MorphMix uses gossiping to discover MIX nodes. Hence, every node is required to know at least one MIX node upfront. The node asks the other MIX node for additional MIX nodes. To establish confidential communication with newly discovered MIX nodes, MorphMix applies a modified Diffie-Hellman (DH) key exchange to establish a symmetric key. This key exchange is assisted by two already known nodes to prevent a MITM attack.

2.3.8 *Anonymizer.com*

Anonymizer Inc. offers a commercial anonymization service¹ via the proxy concept. To forward all TCP/IP traffic via their proxy server, they use a virtual private network (VPN) tunnel between sender and proxy.

As Anonymizer is built on VPN connections and a proxy, it introduces low latency. However, as VPNs usually do not provide padding of messages, incoming messages of the proxy may be linked to outgoing messages.

2.3.9 *JonDo / AN.ON & JAP*

JonDo is another commercial anonymization service² offered by JonDos GmbH. Unlike Anonymizer, JonDo follows the MIX concept. JonDo is based on the open source AN.ON, which again is derived from Java Anon Proxy (JAP) [12]. JAP is a desktop application that acts as a web proxy server and connects to a MIX cascade. The MIX nodes of the cascade are operated by supporters of the AN.ON project, and JonDo respectively.

As JonDo is a high-latency MIX system, the MIX nodes should ensure unlinkability of incoming and outgoing messages. However, due to a request from a law enforcement agency [107], JonDo introduced

¹ <https://www.anonymizer.com>

² <https://www.anonym-surfen.de>

a message flag to make messages linkable across multiple MIX nodes. Still, as the MIX nodes of AN.ON and JonDo are operated by several legal bodies, court orders seldomly address all operators (MIX nodes), and are thus hardly successful [62].

2.3.10 *P5*

P5 is an anonymization service that uses the broadcast concept [134, 135]. However, P5 has not been realized yet and has only been modeled as a packet-level simulation. P5 assumes participants to organize an overlay tree. Within that overlay tree, a sender encrypts a message with the receiver's public key, and broadcasts the encrypted message to all successors in the tree. These successors will then re-broadcast into their subtrees. Furthermore, all messages are padded to the same size of 1KiB.

With P5 nodes can trade anonymity with message overhead: if the sender takes the tree root position, the receiver anonymity set becomes maximal. Likewise, the message overhead becomes maximal as all other nodes will receive the broadcast. This mechanism protects receiver anonymity. To protect sender anonymity as well, P5 nodes generate noise, i.e., packets without meaningful content.

2.3.11 *Summary*

Many of the implementations & services addressed in Section 2.3 show pedigree and have been refined and improved over years. Still, few seem to be successful regarding active users, Tor being the dominant one. The survey from Edman and Yener [48] offers further insights into less known implementations.

Even though Tor is the most popular implementation and has been improved and analyzed countless times—Google Scholar lists 2,639 citations of [44] as of August 20th, 2015—crucial challenges remain. First, Tor was not designed to and cannot protect against colluding anonymity attackers. Second, the application layer should be always considered when designing anonymization services. In particular, strict network layer separation has not worked out well for anonymity in the case of Tor so far. With Tor being located at transport (4) and session (5) layer in the Open Systems Interconnection (OSI) reference model [171] (transport layer (3) in the TCP/IP model), some of Tor's mechanisms are highly influenced by the application layer (7) (layer 4 in the TCP/IP model). Moreover, attacks on the application layer can still violate anonymity. Likewise, other network layers are susceptible to attacks, e.g., as shown by the de-anonymization of originators of Bitcoin transactions [81].

2.4 P2P SYSTEMS

P2P overlay networks are an abstraction on top of an existing network layer, the underlay. Compared to the client-server model, where one

or more nodes take the role of a server and the clients solely connect to such server nodes, P2P omits this role distinction. Instead, all nodes assume both roles and are therefore equal. Usually, P2P systems assume IP as an underlay to establish overlay networks. P2P systems can be classified based on how they establish overlay networks: *unstructured* and *structured* [95]. Some P2P systems, such as Napster [54] and [26, 86], use the client-server model for some functionality and are thus called *hybrid*.

The most important functions performed by a P2P system are *store*, *lookup*, and *retrieve* [95]. The lookup may be combined with retrieval. As lookup does not require much bandwidth compared to store and retrieve, lookup is realized by a central server in a hybrid combination of central and P2P system. This approach speeds up the lookup of information regarding delay and the number of messages.

2.4.1 Structured overlays

When storing information, structured overlays place this information at a location (node) that is determined by some structure. This structure is typically a distributed hash table (DHT). A DHT consists of (key, value) pairs, where the key identifies the information and the value where (which node) to find it. To establish a uniform key representation, typically a hash of the information is used. As the DHT is distributed, no single node is required to keep a complete table. Instead, nodes determine from the key which neighboring peer is closer to the value, i.e., the information. Hence, a progressive localization is possible.

To maintain a structure that allows the determination of the correct neighboring peer, nodes organize themselves in a ring that maintains a predecessor/successor relation among nodes. For that, nodes obtain an ID, typically from the interval $[0, 1)$; each ID represents a position within the ring [148].

The lookup of information via such a ring structure may require traversing half of all nodes in the ring. To speed up the process, nodes may maintain more than two neighbors (predecessor, successor), and thus, introduce shortcuts within the ring—skip lists [67]. A DHT in combination with skip lists provides fast lookup of information with guarantees of the upper bound of nodes to traverse.

2.4.2 Unstructured overlays

Opposed to structured overlays, unstructured overlays store information at random nodes. A lookup, therefore, requires searching for the information in the overlay. Two methods are being used to perform such a search, the graph traversal algorithms *flooding*, and *random walks* [114]. Following the properties of algorithms, flooding and random walk are efficient for looking up popular (highly replicated) information, whereas structured overlays are more efficient for rare information.

FLOODING With flooding, the initiator forwards its request to all neighboring peers. These peers again continue to forward the request to all their peers. Eventually, the request reaches a responder, who discontinues the flooding. To allow the responder to send a message back to the initiator, all forwarding peers are required to store some information about the request for a limited time. Such information can be the tuple consisting of message ID and previous peer.

While flooding is well suited to reach a responder quickly, it causes signaling overhead. To overcome this drawback, the flooded messages contain a time to live (TTL) counter. The initiator initializes the TTL counter with a positive value. Every peer that forwards the message first decrements the TTL counter before sending the message. Once the TTL counter reaches 0, the flooding stops.

RANDOM WALK Random walks can be considered a variation of flooding. Rather than flooding the message to all neighboring peers, with the random walk the message is only forwarded to exactly one randomly selected neighboring peer. The forwarding sequence therefore constitutes a random walk across peers.

Compared with flooding, random walks may cause less signaling overhead. However, the distance between requester and responder may be longer. Furthermore, random walks do not guarantee to reach a responder, even in case a responder is within the TTL counter distance of the initiator.

2.4.2.1 Summary

Summarizing P2P approaches, structured overlays provide the best performance for sparse data, whereas unstructured overlays are best suited for highly replicated data. Structured overlays expose the most information to anonymity attackers as IDs are necessary, and as the attacker can therefore easily target the node with the closest ID to the desired information. Unstructured overlays are therefore best suited for anonymity. The next section introduces network simulations; a method to evaluate how suitable a P2P approach is given a performance metric.

2.5 NETWORK SIMULATION

Simulations are one of the three core methods for evaluating the performance of computer systems [75]. A simulation abstracts from the real systems in the sense that only certain properties considered relevant are modeled in the simulation. Simulations of computer networks are well suited as empirical evaluation are often not possible: a replication of a computer network might not be affordable (the Internet), and evaluation of a productive system may cause interferences. Anonymous communication services are particularly hard to evaluate empirically as they are designed to prevent analysis in the first place [76]. Therefore, network simulations, as well as formal analysis, are the dominant evaluation methods for anonymization services.

For some services, e.g., Tor, specific network simulation tools such as Shadow [76] exist.

This section briefly introduces the concept of discrete event simulation as well as OMNeT++, the discrete event simulator chosen for the quantitative evaluation of this thesis for various reasons as explained below.

2.5.1 Discrete event simulation

A discrete event simulation is based on components emitting and consuming events. The discrete event simulation processes events in chronological order without concurrency (discrete). A discrete event simulator, therefore, stores events within an event queue, sorted chronologically according to the simulation time. An event scheduler takes care of queuing events, deleting events, re-ordering events, and processing events.

The workflow within a discrete event simulation is as follows: the active component emits an event and hands it over to the event scheduler. The scheduler determines at which simulation time the event is going to be processed, and inserts the event at the appropriate position in the event queue. After all actions of the active component are complete, e.g., processing an event, emitting events, and updating its internal state, the scheduler pops the next event from the queue. If the event is scheduled to be processed in the future about the current simulation time, the scheduler advances the simulation time to the processing time. Then the discrete event simulator activates the consuming component, hands over the event, and waits for the component to finish processing.

Discrete event simulations encapsulate the behavior of computer networks. For instance, nodes of a computer network can be represented as components, messages as events, and transmission delay as advancing time.

2.5.2 OMNeT++

Objective Modular Network Testbed in C++ (OMNeT++) [115, 159] is a discrete event simulator written in C++. Other common simulators are OPNET, and Network Simulator 2 & 3 [109]. This thesis considers OMNeT++ the simulator of choice as it is free of charge for academic usage³ and provides many ready to use components from the Internet domain. Moreover, opposed to the other simulators, OMNeT++ has been publicly available including all sources since 1997. OMNeT++ also supports all major operating systems and not only for Unix-based systems (Network Simulator 2,3).

The following sections summarize the structure of OMNeT++, the markup languages, and the add-on framework INET.

³ A commercial version is available under the name OMNEST.

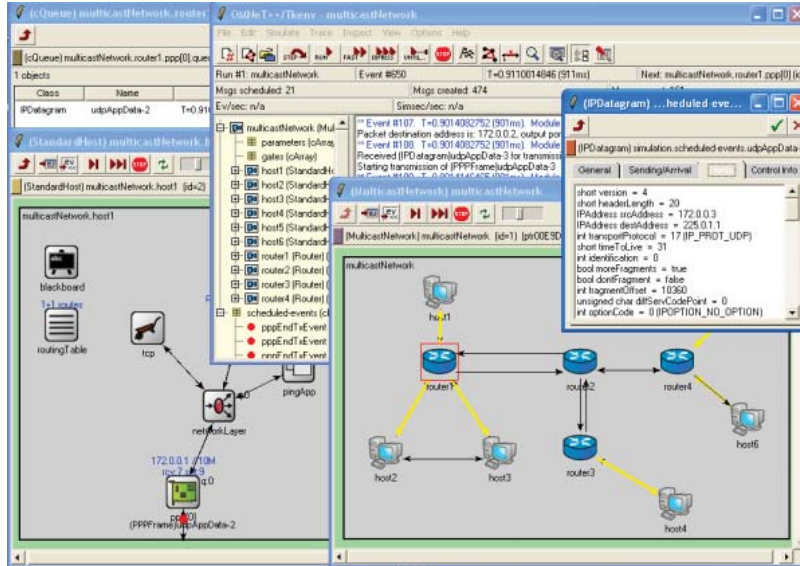


Figure 8: OMNeT++ simulation GUI. [94]

2.5.2.1 Structure

OMNeT++ is structured around the concept of *modules*, much like components in discrete event simulation. These modules can be interconnected and exchange *messages*, the OMNeT++ concept of events. Each connection consists of two *gates*—one for each module—and a module representing the *channel* between these gates. A gate can be incoming, outgoing, and bidirectional. Emitting an event, therefore, corresponds to sending a message out via a gate in OMNeT++. The OMNeT++ simulation GUI is depicted in Figure 8.

OMNeT++ distinguishes two types of modules: simple modules and compound modules. A simple module is a C++ implementation of application-specific functionality. A compound module aggregates one or more modules—simple or compound—gates, and channels into a unit of abstraction. Following the object orientation, modules are typed and inherit from one of OMNeT++’s base classes, e.g., *cModule*, *cSimpleModule*, *cCompoundModule*.

2.5.2.2 Markup languages

While, for instance, compound modules can be implemented in C++ directly, OMNeT++ offers three markup languages to ease simulation development:

NED The Network Description (NED) language describes networks and its elements in OMNeT++. The simplest network can be modeled as two modules with one gate each and a channel connecting both gates. NED can be also used to model modules and channels, including the inheritance of these, their parameters, and gates. The OMNeT++ simulation kernel parses NED files at runtime.

MSG The MSG language describes messages. A MSG file describes the inheritance of a message, e.g., *cMessage* and *cPacket*, as well

as the fields of the message, such as headers and payload. The OMNeT++ toolset translates MSG files into header and C++ files at compile time. During this process, OMNeT++ enriches the messages by additional constants and variables used by the simulation kernel internally.

INI INI files provide the initialization for simulations. In essence, INI files specify or overwrite parameters declared in NED files. The OMNeT++ simulation kernel parses INI files at runtime. INI files are structured into sections called *Config*. Every config describes an individual simulation setup. Such a simulation setup may contain parameter studies, i.e., multiple values of one parameter, and multiple repetitions.

2.5.2.3 INET framework

The OMNeT++ ecosystem consists of manifold frameworks, providing ready to use modules and channels. These frameworks cover specific network simulation areas such as wireless sensor networks, vehicle networks, and INET for Internet components.

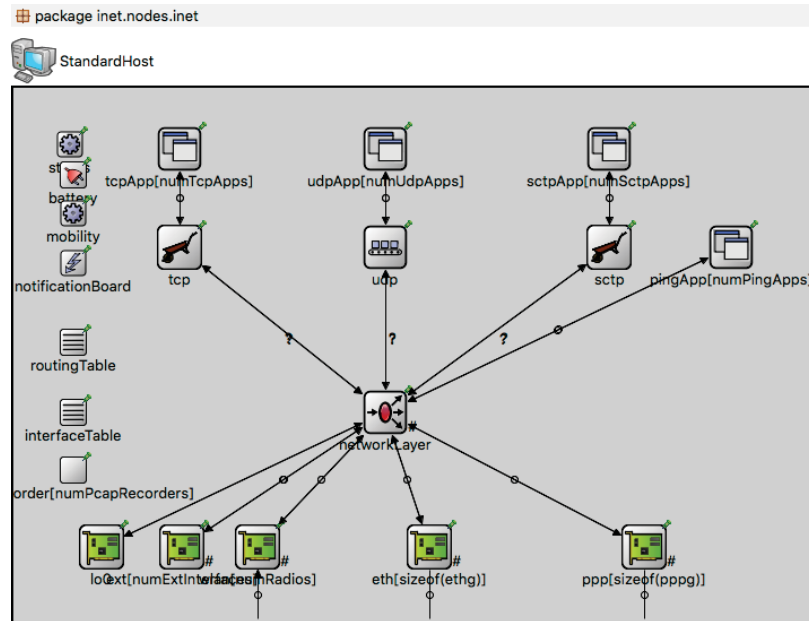


Figure 9: StandardHost compound module of INET, depicted in NED visualization.

INET provides an implementation of a typical Internet connected device, called *StandardHost* as shown in Figure 9. Within this compound module, INET contains module implementations for IPv4 and IPv6 routing, mobility and battery, transport layer protocols such as TCP and UDP, link layer protocol suits such as Ethernet including ARP, IEEE 802.11, and PPP, as well as physical layer channel implementations for these protocols. Further hosts based on these modules are included as well, such as switches, routers, and gateways. Leveraging these components, INET allows building realistic Internet applications on top of this stack.

2.5.3 Summary

In summary, network simulations are well suited to simulate anonymization services. Such simulations can abstract unnecessary details still required for application simulations such as Shadow. OMNeT++ is a mature tool for such simulations and already provides many implemented components from the Internet domain.

2.6 SUMMARY

This chapter has introduced the concepts of Pub/Sub, anonymity, P2P, and network simulation with the goal of establishing the necessary background knowledge that is being used throughout this thesis.

Publish/subscribe is well suited to realize many-to-many applications in a distributed rather than a centralized manner. The paradigm also suits the goal of anonymous communication due to the loose coupling of entities. However, Pub/Sub also requires access to possible confidential information to route this information. Such a lack of confidentiality also poses a threat to anonymity. Thus, anonymous Pub/Sub has to be carefully designed.

Next, this chapter explained the background of anonymous communication. Anonymity can be sub-divided into the four goals unlinkability, sender/receiver anonymity, and relationship anonymity. Three foundational concepts exist to fulfill one or more of these goals: group communication, proxy servers / MIX nets, and DC-nets. Based on these concepts, this chapter briefly surveyed several systems. To measure the anonymity within such systems, a plethora of metrics exist. The metric families of set sizes and entropy have been explained. With the knowledge of concepts and metrics, this Chapter surveyed several well known anonymous communication systems and elaborated on their respective contributions and shortcomings.

Following the insight that P2P systems constitute a common foundation for anonymous communication services, this chapter provided a brief overview of P2P terminology and concepts. This chapter concluded with an introduction into network simulation and the network simulation tool OMNeT++. A network simulation constitutes one of three main evaluation techniques for networked systems, such as Pub/Sub applications, and anonymization services.

REQUIREMENTS AND STATE OF THE ART

The previous chapter introduced necessary background concepts, in particular, [Pub/Sub](#) and anonymous communication services, for this thesis. This chapter analyzes the related work, systems that combine [Pub/Sub](#) and anonymous communication in detail.

To analyze the related work in a structured manner, the following approach is used: a formal anonymous [Pub/Sub](#) system model is introduced in Section 3.1, which allows evaluating requirements for anonymous [Pub/Sub](#) systematically. The requirements for anonymous [Pub/Sub](#) are established in Section 3.2 based on the model.

Anonymity attacker models are mapped to the system model in Section 3.3. These attacker models aim to break the anonymity requirement. Also, this thesis presents novel attack approaches to overcome the main shortcoming of prior anonymity evaluation: existing attacker models are often only formulated on a high abstraction layer, ignore the heterogeneity of [P2P](#), and thus only provide a worst-case but not realistic anonymity assessment. Section 3.4 apply the attacker models to the domain of [Pub/Sub](#) by discussing several methodologies to break anonymity. Moreover, a novel attack that can be applied to unstructured [P2P](#) systems is proposed in Section 3.4.1.

A generic architecture is extracted from related systems in Section 3.5. Based on the architecture, reoccurring technologies for the architectural building blocks are discussed in Section 3.6. Having established and discussed all common elements, sixth, the related anonymous [Pub/Sub](#) systems are analyzed in Section 3.7. Section 3.8 concludes this chapter.

3.1 PUBLISH/SUBSCRIBE SYSTEM FORMALIZATION

This section introduces notation and model for distributed [Pub/Sub](#) systems. This model serves as common ground for the discussion of anonymity attacker models, related work, as well as contributions throughout this thesis. Section 2.1.1 has introduced the concept of [Pub/Sub](#) message dissemination. This section refines this concept by applying a formal notation that also incorporates the concept of anonymization services (cf. Section 2.2) as well as [P2P](#) systems (cf. Section 2.4).

ATTRIBUTES A core property of [Pub/Sub](#) is the loose coupling between participants [51]. Loose coupling is achieved by routing messages according to their content and the interests of participants rather than participant IDs. Section 2.1.1 has introduced common comparison methods of [Pub/Sub](#), i.e., methods to match message contents with interests. The modeling of content and interests is, therefore, a crucial part of the [Pub/Sub](#) formalization.

This thesis uses the concept of *attributes* to formalize message contents and participant interests. Attributes have emerged as an expressive concept in the domain of IT security. Methods of attribute-based access control (ABAC), e.g., eXtensible Access Control Markup Language (XACML) [63], for achieving authorization and attribute-based encryption (ABE) [128] for achieving confidentiality in combination with authorization serve as an example.

To model participant interests, this thesis introduces an attribute set A . Each attribute $a \in A$ reflects a participant's interest. Every participant may be interested in multiple attributes; likewise, messages may be labeled with multiple attributes.

Example 3. *As an example, topic-based Pub/Sub can be directly translated into attributes. Topic x becomes attribute $x \in A$. Subject-based Pub/Sub can be translated by either (i) creating attribute sets out of subjects or (ii) flattening each subject to one attribute. For the first option, a subject $x.y.z$, where x represents the major subject with y being a minor subject, can be modelled as attribute set $\{x, y, z\} \in A$. In this case, the order is lost. For the second option, the subject $x.y.z$ can be represented by attributes representing the valid attribute combinations $\{x, x.y, x.y.z\} \in A$. Type-based Pub/Sub can be modelled via attributes as well by introducing an attribute for every type.*

Finally, content-based Pub/Sub might be modelled via attributes to some extent. Some content-based Pub/Sub implementations, e.g., JMS [41], model "content" as a list of properties attached to every message. Such properties can be directly replaced by attributes.

Social media serve well as an example of attributes: Twitter uses so-called hashtags as labels for tweets (messages). Participants can add hashtags to their tweets, e.g., #TharirSquare. Other participants can then search and even stream (subscription) this hashtag.

GRAPH & OVERLAYS A Pub/Sub system can be modelled as a graph, with computer systems represented by nodes, and with communication links represented by edges. In a centralized Pub/Sub system, one node represents the designated broker, and all other nodes (publishers and subscribers) are connected to this node. This thesis generalizes this topology to a P2P system.

P2P-based Pub/Sub systems, such as [138, 153] [37], distribute the broker functionality among all peers. Peers are interconnected via an underlay, e.g., the Internet. To abstract from the underlay, this model assumes that all peers establish a *basic overlay* via an underlying P2P approach.

The basic overlay is represented as the graph $G = (V, E)$ that incorporates all peers V and interconnects them via overlay edges E . This model assumes that the graph G is connected and unidirectional, i.e., $\forall v_x, v_y \in V$ there is an edge progression from v_x to v_y . As a result, this model can also incorporate underlays without E2E connectivity, such as wireless sensor networks. Each peer v has a set of neighbors defined by adjacent edges. The function $N(v)$ (cf. Equation (11)) returns these neighbors.

ROLES The set of nodes V can be partitioned into publishers \mathcal{P} , subscribers \mathcal{S} , and brokers that are named forwarders \mathcal{F} to follow the **P2P** generalization. Hence, $V = \mathcal{P} \cup \mathcal{S} \cup \mathcal{F}$. A node may take up to all of these *roles* in a **P2P** system. In particular, a node can take the publisher and subscriber role at the same time. Thus $|\mathcal{P} \cap \mathcal{S} \cap \mathcal{F}| \geq 0$ holds.

The roles can be further split according to the attributes. The set \mathcal{P}_a denotes the publishers for attribute $a \in A$. Likewise, \mathcal{S}_a and \mathcal{F}_a denote the subscribers and forwarders for a . The unified set of all nodes involved in handling messages related to a is denoted by V_a .

Following the attribute-specific node notation, the neighborhood function $N(v)$ can be further differentiated in inbound neighbors $N_a^+(v) \subseteq N(v)$ (see Equation (12)) that will forward messages regarding a to the local node v and outbound neighbors $N_a^-(v)$ (see Equation (13)) that node v sends messages regarding attribute a to.

$$N(v) = \{w\} : v, w \in V \wedge ((v, w) \in E \vee (w, v) \in E) \quad (11)$$

$$N_a^+(v) = \{w\} : v, w \in V_a \wedge (w, v) \in E_a \quad (12)$$

$$N_a^-(v) = \{w\} : v, w \in V_a \wedge (v, w) \in E_a \quad (13)$$

MESSAGES Messages in this model are denoted by m . The message m may relate to one or more attributes $A' \subseteq A$ and is written as $m^{A'}$. For simplicity, this thesis uses one attribute $a \in A$, and the message becomes m^a .

A **Pub/Sub** system typically distinguishes three types of messages:

NOTIFICATION A notification, sometimes also called publication, with attribute a contained, is written as m_{notif}^a .

SUBSCRIPTION A subscription for attribute a is written as m_{sub}^a . Subscriptions serve the purpose of setting up routing tables such that all publishers and forwarders on the paths from publishers to subscribers know, in which directions they have to forward notifications.

ADVERTISEMENT If **Pub/Sub** with advertisements is used, the advertisement of attribute a is written as m_{adv}^a . Advertisements aid the subscription process by informing forwarders and subscribers from which directions to expect notifications from.

In combination with the roles: a subscriber $s \in \mathcal{S}_a \subseteq V_a$ subscribes to the attribute $a \in A$ via subscription messages m_{sub}^a and receives notification messages m_{notif}^a originated by the publishers in set \mathcal{P}_a for attribute a . Every publisher $p \in \mathcal{P}_a \subseteq V_a$ may initially advertise the availability of the attribute a via advertisement messages m_{adv}^a that are disseminated in G , e.g., by flooding the neighbors of p . All messages between a subscriber and a publisher may be carried out over multiple hops in G .

ATTRIBUTE OVERLAY The multi-hop dissemination of messages can be considered as another overlay on top of the basic overlay G . Without loops and duplicate paths, such an overlay per attribute forms a mesh $\mathcal{M}_a = (\mathcal{P}_a \cup \mathcal{S}_a \cup \mathcal{F}_a, E_a)$ that includes all publishers \mathcal{P}_a , all subscribers \mathcal{S}_a and depending on the specific **P2P-based Pub/Sub** approach it will also include a certain number of nodes, so-called *forwarders* \mathcal{F}_a , that are not interested in attribute a at all, but will perform brokering in between publishers and subscribers. E_a (cf. Equation (14)) denotes the set of edges that connects the nodes in \mathcal{M}_a . This is the most basic definition of overlay edges. A **Pub/Sub** system may remove edges and introduce new edges. Nodes in \mathcal{M}_a can be summarized as $V_a := \mathcal{P}_a \cup \mathcal{S}_a \cup \mathcal{F}_a$.

$$E_a = \{(n_1, n_2) \in E : n_1, n_2 \in V_a\} \quad (14)$$

$$path_a(p, s) = \begin{cases} \{p, s\} & \text{if } (p, s) \in E_a \\ \{p\} \cup path_a(x, s) & \text{if } (p, x) \in E_a \wedge path_a(x, s) \neq \emptyset \\ \emptyset & \text{else} \end{cases} \quad (15)$$

$$\forall p \in \mathcal{P}_a \forall s \in \mathcal{S}_a : path_a(p, s) \quad (16)$$

The function $path_a(p, s)$ (cf. Equation (15)) describes the overlay path in \mathcal{M}_a between nodes p and s . The function returns this path as a node progression. With the assumption of no loops and no duplicate paths in \mathcal{M}_a , there exists at most one path between two nodes. To ensure the requirements of subscription integrity and service integrity (cf. Section 2.1.2), there exists a path between every publisher and every subscriber as defined by Constraint (16).

TIME & CHURN A **Pub/Sub** system can be also subject to churn over time and thus inherently dynamic. The time $t \in T : T = \{1, 2, \dots\}$ indicates snapshots, e.g., G^t and \mathcal{S}_a^t . Every system snapshot is represented by an increment of t .

Node churn $\varphi \in [0, 1]$ denotes the ratio of nodes of the total node population V that are subject to churn during a time interval $[t_i, t_{i+1}]$. Three variations of churn are considered:

- With φ_{join} , $\varphi_{join} \times |V|$ nodes join the basic overlay G during the time interval $[t_i, t_{i+1}]$.
- With φ_{leave} , $\varphi_{leave} \times |V|$ nodes of the basic overlay G leave during the time interval $[t_i, t_{i+1}]$.
- With φ , this ratio is distributed among φ_{join} and φ_{leave} . That means, $\varphi_{join} = \frac{\varphi}{2}$ and $\varphi_{join} \times |V|$ nodes join during the time interval; $\varphi_{leave} = \frac{\varphi}{2}$ and $\varphi_{leave} \times |V|$ nodes out of the nodes that are already part of G at time t_i leave within the interval.

SYMBOL	EXPLANATION
A	Set of attributes
$G = (V, E)$	Graph of the basic overlay
V	All nodes / peers / participants
E	Direct connections between nodes
$N(v)$	Shortcut function for neighbors of v
\mathcal{P}_a	Publishers with attribute a
\mathcal{S}_a	Subscribers for attribute a
\mathcal{F}_a	Forwarding peers for attribute a
$\mathcal{M}_a = (V_a, E_a)$	Overlay mesh for attribute a
m_{notif}^a	Notification message with attribute a
m_{sub}^a	Subscription message for attribute a
m_{adv}^a	Advertisement message for attribute a
$N_a^+(v) \subseteq N(v)$	Neighbors sending m_{notif}^a to v
$N_a^-(v) \subseteq N(v)$	Neighbors v sends m_{notif}^a to
$path_a(p, s)$	Overlay path from p to s for attribute a
$t \in T$	The time or snapshot t of the system

Table 1: System formalization

The churn rate ϕ remains constant over time intervals. Churn occurs in two characteristics: first, nodes join and leave G , which also affect overlays \mathcal{M}_a . Second, subscribers and publishers join and leave attribute overlays \mathcal{M}_a but not G . Nodes send a subscription message (m_{sub}) to join an attribute overlay. Advertisement messages m_{adv} from publishers indicate the availability of overlays. To leave an attribute overlay, they have to send an unsubscription (m_{unsub}) or unadvertisement (m_{unadv}), depending on their role in the attribute overlay. When nodes fail, their neighbors act as if they received an unsubscription / unadvertise message from that node.

SUMMARY Table 1 summarizes the introduced notation. This notation describes two network abstraction layers of a Pub/Sub system, the basic overlay and the attribute overlay. Furthermore, the notation describes participants and their roles, messages exchanged in such a system, and churn. The notation does not explain how \mathcal{M}_a is constructed out of G . This overlay creation is part of the respective Pub/Sub systems.

3.2 ANONYMITY REQUIREMENTS FOR PUB/SUB

Security requirements for Pub/Sub have been introduced in Section 2.1.2. This section elaborates on requirements particular to anonymity and puts them into perspective the formal model established in the previous section.

Xiao [168] enumerates *publishing anonymity*, *sending anonymity*, and *receiving anonymity* as anonymity goals. Publishing anonymity requires that the information can be created without the creator being discovered. Sender and receiver anonymity have been discussed in Section 2.2.1 and require that a message is not linkable to a sender or receiver.

Further requirements exist [113]: the combination of sender and receiver anonymity leads to relationship anonymity, i.e., the requirements of sender and receiver not being linked together. Unlinkability requires no two messages to be linked together.

The less differentiated term *privacy* is also being used [118, 162] in the context of Pub/Sub. It is important to note that this requirement originates from the insight that anonymity should be considered in combination with confidentiality [129]. A violation of either requirement can easily lead to the violation of the other one as well:

ANONYMITY WITHOUT CONFIDENTIALITY Anonymity without confidentiality means that data or messages are available in plaintext, but the sender and receiver are anonymized. Prior research has shown that profiling and correlation are possible, resulting in a potential de-anonymization of sender and receiver [105].

CONFIDENTIALITY WITHOUT ANONYMITY Confidentiality without anonymity means that data or messages are linked to sender and receiver, but the plaintext of the data or message is not available.

Example 4. One example of the anonymity without confidentiality issue is the transmission of unencrypted information, or information that contains metadata, with an anonymization service. The amnesic incognito live system (Tails) for instance explicitly warns¹, that although Tails uses Tor to anonymize the communication metadata, application metadata such e-mail addresses are not protected.

An example for confidentiality without anonymity is the non-anonymous but encrypted access of web pages. Even if the information transmitted via the HTTPS protocol is encrypted, the communication destination, e.g., the Internet protocol (IP) address and thus an associated regime-critical blog, reveals the potentially transmitted information.

This thesis transforms the terms sender and receiver anonymity into *publisher anonymity* and *subscriber anonymity* to better fit Pub/Sub. That means, given a Pub/Sub system including all exchanged messages:

1. An anonymity attacker cannot link any of the messages— m_{notif}^a , m_{sub}^a , m_{adv}^a —to a publisher or subscriber.
2. An anonymity attacker cannot link any attribute $a \in A$ to a publisher or subscriber.

¹ <https://tails.boum.org/doc/about/warning/>

The second requirement follows from the observation that confidentiality and anonymity must be considered in combination [129]. To measure the anonymity—the inverse success of an anonymity attacker—, this thesis uses the two metrics *anonymity set size* [112] and *degree of anonymity/entropy* [43] (cf. Section 2.2.3). With these metrics, anonymity can be formalized as follows:

SUBSCRIBER ANONYMITY Every subscriber can hide in a reasonably large anonymity set $anonSet_{adv}$ from the perspective of an anonymity attacker. The attacker attempts to establish the minimal anonymity set containing only subscribers, and thus disclosing these subscribers, for a given attribute a from a set of attribute \mathcal{A} . Thus, a Pub/Sub system meets subscriber anonymity w.r.t. to an adversary \mathcal{A} , and a set size k , if Equation 17 holds, i.e., the attacker cannot reduce any anonymity set below the size of at least k participants. Using anonymity degree as metric, the system provides anonymity if Equation 18 holds, the attacker \mathcal{A} cannot learn more than threshold T about the system (cf. Equation (4) on page 20 for the definition of d).

$$\forall a \in \mathcal{A} : |anonSet_{\mathcal{A}}(a)| \geq k \quad (17)$$

$$\forall a \in \mathcal{A} : d_{\mathcal{A}}(a) \leq T \quad (18)$$

PUBLISHER ANONYMITY Similar to subscriber anonymity, the Equations (17), (18) can be applied to publisher anonymity as well.

SUMMARY This section established the terms subscriber and publisher anonymity. Furthermore, the two metric anonymity set size and anonymity degree were chosen to measure the anonymity. The conditions if publisher and subscriber anonymity hold are based on an anonymity attacker \mathcal{A} . The next section will elaborate on models for this attacker.

3.3 ATTACKER MODELS FOR ANONYMITY

Manifold anonymity attacker models have been proposed, differing in their capabilities and the attack approach. Dingledine et al. [44] discuss capabilities extensively, although their anonymization service *Tor* is only designed to protect against a subset of these capabilities. Compared to capabilities, attack approaches have been less extensively studied. Attacks, e.g., the ones studied by the anonymity metrics discussed in Section 2.2.3 (page 18), are evaluated via theoretical worst-case capabilities.

This section summarizes existing attacker models in Section 3.3.1 and elaborates further on two attacker models in the context of the system formalization in Section 3.3.2. These attacker models describe the capabilities of anonymity attackers. The following Section 3.4 will then elaborate on how these capabilities can be applied to break anonymity in Pub/Sub systems.

3.3.1 Properties & capabilities

Edelman and Yener [48] summarize the attacker properties used to evaluate anonymization services as *capability*, *visibility*, *mobility*, and *participation*. These properties can be adapted as follows to the domain of anonymous Pub/Sub:

CAPABILITY The capability defines what actions the attacker can take. A *passive* attacker eavesdrops and monitors; an *active* attacker extends the passive attacker with the Dolev-Yao model [46]. The active attacker can therefore delay, drop, duplicate, modify, recombine, and generate messages. The passive attacker is particularly strong in Pub/Sub when it takes a forwarder role $f \in \mathcal{F}$ as it may observe many messages. The active attacker is powerful in the publisher and subscriber roles as it may stimulate the other roles to react.

VISIBILITY The visibility or scope defines to which extent the attacker can eavesdrop the interaction with a Pub/Sub system. This scope can be defined as a subset of nodes $V' \subseteq V$ and edges $E' \subseteq E$. Two types of visibility are commonly used as attacker model: the *global* attacker can observe or interact with all edges E . The *local* attacker is restricted to one node $v \in V$ and the set of adjacent edges E' as defined by Constraint (19).

$$\forall w \in V, e = (v, w) \in E' : w \in N(v) \wedge (v, w) \in E \quad (19)$$

In the case of $|V'| > 1$, the attacker can be also called *colluding*.

MOBILITY The mobility defines if the visibility of the attacker is invariant over time. The *static* attacker observes and interacts with the same part of the system over time, i.e., $\forall t \in T : V'^t = V'^0 \wedge E'^t = E'^0$. The *adaptive* attacker chooses always the visibility that suits best. For instance, P2P systems that allow IDs to be changed can be used for the adaptive attacker.

PARTICIPATION The participation distinguishes if the attacker is *internal*, i.e., part of the system, or *external*. An internal attacker takes the visibility of nodes $V' \subseteq V$. It, furthermore, has access to secrets and key material of these nodes. The external attacker typically takes the visibility of edges $E' \subseteq E$ and thus lacks access to secrets.

Literature in the domain of anonymous Pub/Sub often refers to the *honest but curious attacker* [27, 103, 118]. This attacker has the properties passive, local/global, static, and internal. This attacker often takes the role of the only forwarder (broker) in the system, and thus, the visibility covers all edges $E' = E$. Therefore despite having a visibility of only one node, the attacker can be considered global as well.

3.3.2 Selected attackers

This thesis uses two attacker models (property combinations): the *global observer* $G\mathfrak{A}$ and the *malicious insider* $I\mathfrak{A}$. Both attackers describe realistic attacker properties—the $G\mathfrak{A}$ large-scale US National Security Agency (NSA) like eavesdropping and the $I\mathfrak{A}$ compromised computing devices. Both attackers are also used in combination. Also, the malicious insider is also used with the collusion of multiple nodes.

GLOBAL OBSERVER $G\mathfrak{A}$ The goal of this passive, global, static, and external attacker is to break publisher and subscriber anonymity by eavesdropping on all edges und thus observing all messages.

Example 5. *One scenario for this attacker is intelligence organizations attempting to obtain network IDs of online social network (OSN) users communicating about a certain attribute, e.g., monitoring the message flow of many Internet service providers (ISPs) to determine IP addresses of micro-bloggers using a certain hashtag. The Snowden leaks discussed in the media suggest espionage programs capable of eavesdropping at ISP and Internet exchange level. It is, therefore, reasonable to assume that ISPs and Internet exchanges are capable of performing such monitoring by themselves.*

This attacker remains passive as it does not manipulate messages. It is external as it does not possess (a priori) secrets or key material. The attacker is furthermore global as it can observe the full communication network (E). From the global property follows that this attacker is also static, and there is no need for adaptivity. According to the system model (cf. Section 3.1), the attacker observes notifications m_{notif} , knows an attribute a without associated secrets, possess topology information of the basic membership management $G := (V, E)$, and attempts to learn S_a or P_a . That is, break subscriber or publisher anonymity.

The attacker can achieve this goal for instance via the statistical disclosure attacks [31, 33, 158]. Here, the attacker discloses message sinks, as well as nodes that remain persistent in \mathcal{M}_a after overlay restructuring, as subscribers.

MALICIOUS INSIDER $I\mathfrak{A}$ The goal of this active, internal, and local attacker is to break publisher and subscriber anonymity by participating in the anonymous Pub/Sub system and exploiting the protocol. Depending on the protocol, this attacker is static or adaptive. Furthermore, this attacker can be colluding with other malicious insiders as well as with the global observer.

Example 6. *One example for this attacker is one or more compromised nodes in the control of criminals, e.g., computers that have been infected with trojans and thus, expose key material and connectivity to the attacker.*

This attacker is active and therefore modifies and suppresses messages at will, decrypts, and forges valid messages with key material for an attribute a , and colludes with several attacking nodes. In combination with the global observer, this attacker can also access the

topology information G as well as observe messages. An attack using this model has been published in [37]. To formalize collusion of malicious insiders, the system formalization in Table 1 is extended by the set of colluding attackers $\mathcal{C} = V' \subseteq V$.

SUMMARY This section has briefly summarized common anonymity attacker properties. Two models were discussed in detail: the global observer $G\mathcal{A}$ is a passive observer of the whole Pub/Sub system; the malicious insider $I\mathcal{A}$ takes one or more nodes (collusion) and actively attempts to break anonymity. The next section introduces a novel and concrete anonymity attack approach based on the combination of both models.

3.4 ATTACKS ON ANONYMOUS PUB/SUB

Following the attacker capabilities presented in the previous section, a multitude of attack approaches are possible to break anonymity in Pub/Sub. This section structures the attacks on anonymization services presented in the previous chapter (Section 2.3.6.2, page 23) according to their type, and explains how the attack can be applied to Pub/Sub. Afterwards, a novel attack on anonymous Pub/Sub that exploits timing information is proposed in Section 3.4.1.

STATISTICAL DISCLOSURE Statistical disclosure [33] leverages the information gained over prolonged periods of information. This type of attack assumes that over time, true receivers handle more messages than pure forwarders. Applied to Pub/Sub, the anonymity attacker assumes that subscribers receive more notifications than pure forwarders. Alternatively, the attacker may assume that subscribers remain as members of an attribute overlay for more time intervals than pure forwarders. This may happen due to churn, i.e., subscribers reconnect to the attribute overlay in case of leaving neighbors, but pure forwarders may not.

To mount this type of attack, the global observer is required. An attacker with the capabilities of the global observer model traces the flows of notification. Given the trace of a notification flow, the attacker knows which nodes are part of this flow, and are therefore member of an attribute overlay. By tracing multiple notifications of the same attribute overlay, the attacker gains the necessary statistical information. Malicious nodes can be used to enhance this attack: malicious insiders attempt to become pure forwarders, and then interrupt connections to neighbors.

TOPOLOGY ANALYSIS Topology analysis uses the topology of attribute overlays to reason about the role of nodes. This type of attack assumes that the topology of an attribute overlay follows a deterministic construction. As before, the anonymity attacker traces the flow of a notification and identifies the involved nodes as well as their connections. Given this topology and knowledge about the overlay

construction method, the attacker may infer that a publisher is likely to be located in the center of such a topology. Likewise, subscribers may be located in leaf positions. To mount this type of attack, the global observer is required as well.

GRAYHOLING Grayhole or eclipse attacks concentrate traffic in malicious nodes. This type of attack assumes that the anonymity attacker can increase his neighborhood or steal traffic by some other means, e.g., by gaining popularity. The attacker may then reason that its neighbors in an attribute overlay are likely to be publishers and subscribers—as the attacker steals more traffic, less pure forwarders can be part of the attribute overlay. The attacker may also use the grayhole attack as an aid for another attack, such as the statistical disclosure or the topology analysis. To mount this type of attack, the capabilities of a malicious insider are required.

TIMING ATTACKS Timing attacks exploit the reaction time of nodes. This type of attack assumes a difference in the reaction time of pure forwarders and publishers and subscribers. The attacker either acts itself, e.g., by sending a request and waiting for a response, or measures the reaction time of nodes given observable actions. A node that reacts immediately to an advertisement may be considered a subscriber, whereas a node that reacts delayed or not at all may be considered a pure forwarder. A subscriber may also forward notifications slower than a pure forwarder, as the subscriber also processes the content of notifications, whereas the pure forwarder does not.

To mount this type of attack, the capabilities of a global observer are required. A malicious insider may also analyze its neighbors or aid the global observer. The following section presents a novel anonymity attack that belongs to the category of timing attacks.

3.4.1 *A timing attack in anonymous communication*

This section presents a timing attack published in [38] to break subscriber anonymity. This attack exploits the principle that many anonymous communication systems are based on request/response semantics in their protocols. These request/response semantics can be used by the malicious insider $\mathcal{I}\mathcal{A}$ in collusion with the global observer $\mathcal{G}\mathcal{A}$ to de-anonymize responders—subscribers in the context of Pub/Sub. The remainder of this section is structured as follows: first, the attack approach is presented and formalized in Section 3.4.1. Second, methods to determine delays are discussed in Section 8.

This attack is based on the assumption that the attacker can send request messages regarding an attribute a . Then, a target node, which the attacker desires to de-anonymize, responds, and the attacker reasons about this node's position in the basic overlay G . That means, the attacks aims to identify nodes related to a , e.g., the set of subscribers \mathcal{S}_a in a Pub/Sub systems. This attack can be applied to various types

of anonymous communication systems, e.g., the anonymous [Pub/Sub](#) system [37] and Freenet [24]

This attack assumes that both attackers, $G\mathfrak{A}$ and $I\mathfrak{A}$ collude. The $G\mathfrak{A}$ provides topology information about the anonymous communication system, the graph G . The $I\mathfrak{A}$ provides secrets and key material about α , and can send messages. With both attackers colluding, the $G\mathfrak{A}$ can furthermore trace message flows related to α and reduce G to the attribute mesh \mathcal{M}_α .

For this attack to work, it is also important that the analyzed system does not impose artificial message delay, i.e., that the dominating factors in the message transmission delay are node and edge capability and load. Also, the system must provide some option to estimate or measure such delays.

ATTACK APPROACH To perform the attack, the $I\mathfrak{A}$ sends a request into G , awaits a response, and measures the round-trip time ([RTT](#)). Given the attacker can estimate average delays between nodes, the attacker can establish a candidate set (anonymity set) of potential responders within the attribute mesh \mathcal{M}_α provided by the $G\mathfrak{A}$. The major challenges for a realistic implementation of this approach are:

1. How can the attacker derive an anonymity set/entropy from the [RTT](#)?
2. How can the attacker model and refine the anonymity set/entropy over multiple iterations?

Regarding the first questions: given the stated assumptions hold, after sending a request from the $I\mathfrak{A}$, the attacker will receive zero or more responses. For the remainder of this explanation, one response is assumed. Between request and response, the attacker observes the [RTT](#) denoted as δ . Given that the transmission delay is equal for every edge in \mathcal{M}_α , the attacker can estimate the number of edges d . Given the number of edges, the attacker can establish the anonymity set of nodes at the appropriate distance, i.e., nodes with $v \in V_\alpha : |\text{path}(I\mathfrak{A}, v)| = d$.

Example 7. For a [Pub/Sub](#) system that requires advertisements, the request/response behavior can be modelled as an advertisement for requests, and subscriptions for responses. For [Pub/Sub](#) via unstructured [P2P](#), it is necessary to distribute advertisements before subscriptions are sent. Without advertisements, every node would have to setup routing tables such that a notification from every node can be forwarded to every subscriber. That would cause storage overhead for every node. That means a subscriber will subscribe once it received an interesting advertisement. The [RTT](#) δ can then be modelled as given in Equation (20).

$$\overbrace{\delta(m_{adv}^x, m_{sub}^x)}^{\text{observable}} = \overbrace{\sum_{j=0}^{d-2} \delta(m_{adv}^j, m_{adv}^{j+1})}^{\text{advertisements}} + \overbrace{\sum_{j=d-2}^0 \delta(m_{sub}^{j+1}, m_{sub}^j)}^{\text{subscriptions}} \quad (20)$$

The left-hand part of Equation (20) is observable by the \mathcal{IA} , the right-hand part describes what happens inside the Pub/Sub system. The symbol d denotes the distance or path length between the attacker and the responder. However, as this distance is on the right-hand side, the attacker cannot observe it.

Function $\delta(m_w^x, m_v^y)$ measures the time in between sending message m_w from node $v_x = \mathcal{IA}$ and receiving m_v at node v_y .

As an alternative to this example, subscriptions and notifications can be used as well. Assuming a high notification rate, a subscription can be considered as the request, which is answered by the next notification.

Following up on the example, to obtain d from Equation (20), the attacker needs an estimate for the delay components δ on the right-hand side of the equation. These delay components can be approximated via an average delay δ_{avg} that fulfills Equation (21).

$$\delta_{avg} = \frac{\sum_{j=0}^{d-2} \delta(m_{adv}^j, m_{adv}^{j+1})}{d} \quad (21)$$

Given that the attacker obtains such an average delay, it can approximate the distance d as in Equation (22). The division by 2 is necessary as $\delta(m_{adv}, m_{sub})$ describes the RTT. However, the attacker aims at obtaining the one-way distance to the responder, and not the round-trip distance.

$$d \approx \frac{\delta(m_{adv}, m_{sub})}{2 \times \delta_{avg}} \quad (22)$$

With d given, the attacker can rule out nodes closer than d , as these nodes would have responded earlier. That is, all nodes $v \in V_a : |path(\mathcal{IA}, v)| < d$ are not in the anonymity set. However, the attacker cannot rule out nodes further away, i.e., $v \in V_a : |path(\mathcal{IA}, v)| > d$. The responder at distance d may suppress responses further away than d . Thus, the attacker may not observe responses from those nodes, even those nodes relate to a and, therefore, should be de-anonymized.

This behaviour is a challenge for the attacker when modeling the attack results via an anonymity set: nodes at distance d are likely to be responders, nodes closer than d are impossible to be a responder. Hence, the attacker adjusts his initial probability distribution \mathbf{v} . Equation (24) shows how probabilities are set after an attacker, i.e., after the attacker established d . Here, a probability of 0 means that a node is not in the anonymity set.

$$\mathbf{v}_1 = (0, 0, 0, \frac{1}{3}, \frac{1}{5}, \frac{1}{6}, \frac{1}{5}, \frac{1}{20}, \frac{1}{20}) \quad (23)$$

Example 8. Figure 10 shows an example graph G with nodes $v_1, \dots, v_9 \in V_a$. Nodes v_4, v_5, v_7 are the nodes \mathcal{IA} attempts to expose. At this stage (1), the probability distribution the attacker set up is given in Equation (23).

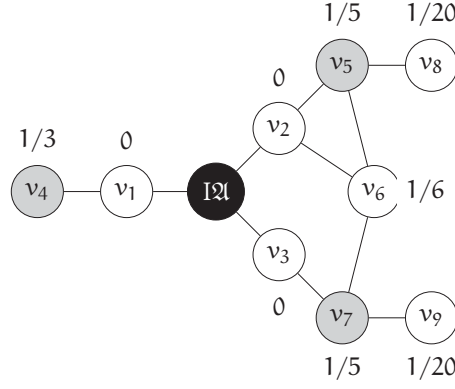


Figure 10: Example graph with node IDs and probabilities.

$$\mathbf{v}[v_x] = \begin{cases} 0, & \text{if } v_x \text{ closer than } d \text{ or } v_x \in \mathcal{C}_a \text{ (j times)} \\ \frac{\lambda}{(k+1)}, & \text{if } v_x \text{ at distance } d \text{ (k times)} \\ \frac{\phi}{(l+1)}, & \text{if } v_x \text{ further than } d \text{ (l times)} \end{cases} \quad (24)$$

The equation shows how the attacker can eliminate j out of $|V_a|$ nodes that are closer than d hops on the shortest path in basic overlay G . Hence, all remaining (k at distance d , l at a distance greater than d) nodes get a higher probability. The parameters λ and ϕ model how the probabilities are distributed among likely nodes at distance d and possible nodes further away than d . The parameter λ should be set to $\lambda \geq 1.0$ and the parameter $\phi \leq 1.0$ solved in accordance (cf. Equation (26)).

Interpreting Equation (24), every node v_x with $\mathbf{v}[v_x] > 0$ is in the anonymity set. Using this approach, the attacker can derive anonymity sets from the RTT. Following up on the second question, the attacker can refine the probability distribution over multiple iterations via four steps:

1. A node v_x which probability has been set to 0 remains at 0.
2. A node v_x with a positive probability that is closer than d is set to 0.
3. The probabilities of nodes at or further away than d are added.
4. The probability distribution is normalized.

The attacker sets up an initial probability distribution. Equation (25) provides such an initial assignment. Here, the probabilities are assigned based on an estimation of the total number of responders—for Pub/Sub the set of subscribers for attribute a given as \mathcal{S}_a . The set \mathcal{C}_a models the malicious insiders \mathcal{ID} . Constraint (26) ensures that the distribution can be normalized in step 4.

$$\forall v \in V_a : \mathbf{v}[v] = \begin{cases} 0 & \text{if } v \in \mathcal{C}_a \\ \frac{|\mathcal{S}_a|}{(|V_a| - |\mathcal{C}_a|) * |\mathcal{S}_a|} & \text{else} \end{cases} \quad (25)$$

$$\sum_{v_i \in V_a} \mathbf{v}[v_i] = 1 \quad (26)$$

ESTIMATING THE AVERAGE DELAY The malicious insider depends upon the average delay δ_{avg} to calculate the distance d . Approaches estimating this delay exist [66], but depends upon additional protocols such as DNS that do not incorporate the application layer delays of anonymization services. The cryptography used by anonymization service may introduce significant delay. This section, therefore, proposes three strategies to obtain average delay estimates.

NEIGHBOR RTT The $\mathcal{I}\mathcal{A}$ can use its neighbors $N(\mathcal{I}\mathcal{A})$ to extrapolate the average delay. Assuming that the attacker can send messages to the neighbors that will trigger an immediate response, the attacker can measure the **RTT** and estimate $\delta_{avg} \approx \text{RTT} \div 2$.

Some basic overlay protocols provide the necessary means for such immediate responses. For instance, the last packet of a TCP window and the following acknowledgement; neighborhood requests as part of gossiping; heartbeats in conjunction with responses. Having multiple neighbors, the attacker can establish the average delay over all neighbors as given in Equation (27).

$$\delta_{avg} = \frac{\sum_{v \in N(\mathcal{I}\mathcal{A})} \delta(m_{request}, m_{response})}{|N(\mathcal{I}\mathcal{A})|} \quad (27)$$

While the requirements for the anonymization service for this strategy are low, it is biased by the connectivity of $\mathcal{I}\mathcal{A}$.

HOP COUNT A strategy that reduces the bias of the $\mathcal{I}\mathcal{A}$ uses a hop counter of messages in conjunction with loops in G . For that, the attacker sends a message with a hop counter, waits until it receives the message again (loop), and divides the **RTT** by the hop count difference. Typical instances of (inverse) hop counters are **TTL** counters. Such counters are used by flooding algorithms, and random walks. For instance, search requests in Freenet use a **TTL**. Another example is advertisement messages in [37]. Alternatively, the global observer might trace the path of a message and can thus observe the length of the path without a hop counter.

This strategy reduces the bias of the $\mathcal{I}\mathcal{A}$. However, is also imposes additional requirements on the anonymization service. Furthermore, this strategy is less stealthy as the attacker has to send messages compared to deriving information from already existing messages. Hence, the attacker may expose itself.

COLLUSION Collusion is an extension of the hop count strategy. Rather than relying on loops and [RTT](#), multiple colluding $I\mathcal{A}$ s can measure the delay on a path in between them as given in Equation (28).

$$\delta_{avg} = \frac{\delta(m^{I\mathcal{A}_1}, m^{I\mathcal{A}_2})}{|\text{path}_a(I\mathcal{A}_1, I\mathcal{A}_2)|} \quad (28)$$

This strategy does not rely on loops and can be refined by adding more colluding attackers. Furthermore, assuming that both attackers are on a path between a requester and responder, the attackers may derive the transmission delay from existing messages rather than generating new messages. Thus, this strategy is potentially stealthier than the hop count strategy.

SUMMARY This section presented a novel anonymity attack that makes use of a malicious insider and a global observer. The attack aims to create anonymity sets for responders given an attribute a . The attack furthermore refines and reduces the anonymity set over multiple iterations. Also, malicious insiders can collude to provide more iterations. The attack can be applied to anonymization services that build upon a request/response semantic, e.g., search in [P2P](#) file sharing, and advertisements with subscriptions in [Pub/Sub](#).

The attack also requires some delay information about the basic overlay besides the request/response semantic. Therefore, three methods to obtain delay estimation via the malicious insider were proposed as well.

The next section structures related anonymous [Pub/Sub](#) systems into common building blocks, i.e., following the system model, and discuss these building blocks.

3.5 STRUCTURING SYSTEM ARCHITECTURES

This section introduces an architecture that splits anonymous [Pub/Sub](#) into so-called *building blocks*. Then, common solutions for these building blocks are discussed. First, the formation of building blocks is explained along the roles participants can take. Second, the formation of building blocks along the lifecycle phases of a system are discussed.

Anonymity is often considered [[102](#), [118](#), [162](#)] as an *add-on* requirement that can be fulfilled by an anonymization service (cf. Section [2.2.2](#), page [16](#)) such as Tor [[44](#)]. This approach protects communication metadata, e.g., [IP](#) addresses, from exposure towards anonymity attackers.

Applications running on top of such a service, e.g., [Pub/Sub](#) may introduce additional metadata that circumvents the anonymity protection as shown in Figure [11](#). For instance, attacks on web browsing as an application in conjunction with Tor have been proposed (cf. Section [2.3.6.2](#), page [23](#)). These attacks use metadata leaked by the web

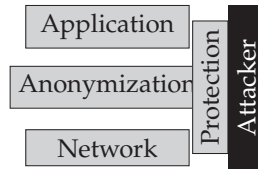


Figure 11: Anonymization service cannot full the application from exposing information to the attacker.

protocol hypertext transfer protocol ([HTTP](#)) to de-anonymize users [14, 69] and break even confidentiality, e.g., by revealing visited web sites [1, 68]. A lesson learned from these attacks is that anonymity should be considered throughout the whole technology stack of an application. The *Privacy by Design* initiative [71] postulates this approach as well. This section proposes an architecture that splits systems into manageable building blocks. While anonymity has to be considered for every building block, designing and analyzing building blocks with one specific functional goal each is easier and less error-prone as with one monolithic system.

DESIGN CONSIDERATIONS A particular challenge for anonymous [Pub/Sub](#) arises from conflicting goals between anonymization services and [Pub/Sub](#)—the first one attempts to distribute message routing among many nodes to protect anonymity, and the latter attempts to centralize routing functionality to increase efficiency.

Within the traditional [Pub/Sub](#) paradigm, which still reflects the typical usage of [Pub/Sub](#) middleware today, e.g., [JMS](#) [41], a broker acts as a central entity [51] and controls communication; participants just state interests and provide content. Compared, anonymization services such as Tor and Crowds empower the participant to organize communication, e.g., build Tor circuits, and thus take control over the infrastructure. Anonymous [Pub/Sub](#), therefore, must balance both extremes, preferably in an adjustable manner. Thus, the architectural goal of this thesis is a solution that maintains a balance in-between both concepts. Every participant must be able to manage her anonymity, i.e., control privacy enhancing technologies (PETs) and measures the current level of protection, whereas the infrastructure may only take some control to maintain requirements such as integrity and availability.

Example 9. *Anonymity may conflict with availability if a malicious user abuses the system by spamming other participants with messages. These participants may become unavailable due to message overload. To counter this issue, the malicious user must be detected and isolated from the system. For that, correlation of messages to users is required. However, correlation of messages violates unlinkability. As a consequence, the infrastructure requires some control to correlate messages and protect availability. In return, some control over anonymity is taken away from the user.*

This thesis follows the *Privacy by Design* principle 3: “Privacy Embedded into Design”. Therefore, anonymity as an aspect of privacy has to be considered for every component of the architecture. This

approach minimizes issues that arise from add-on anonymity solutions like Tor [44] due to the missing consideration of application layer leak of information. Furthermore, this thesis follows the design principles of “high cohesion and loose coupling” [18] and applies it to the components. High cohesion, on the one hand, requires that elements of one building block highly depend on and interact with each other. Loose coupling, on the other hand, reduces interdependencies between building blocks. These principles support exchangeable building blocks—depending on the selection and prioritization of requirements. This structure should, therefore, ease the selection of building block technologies for applications and their respective requirements.

Example 10. *If one building block encapsulates the direct communication between participants, this building block also defines how neighbors are discovered, and a connection is established. With add-on anonymity protection, the process of neighbor discovery would be left out from anonymity analysis.*

OVERVIEW OF BUILDING BLOCKS The overview and dependencies of all six building blocks are depicted in Figure 12. The diagram presents the building blocks of one node participating in the Pub/Sub system (see Figure 13).

- *Basic membership* establishes and maintains the connection of each participant with the system.
- *Attribute localization* allows subscribers to find attributes offered by publishers and is thus used by both roles.
- *Matching* covers the functional Pub/Sub requirement (cf. Section 2.1.2) to maintain routing tables to route messages. This block applies to the forwarder role.
- *Community management* repairs and optimizes overlay networks whenever topological changes occur. This block applies to forwarders, too.
- *Content distribution* distributes notifications to subscribers. This block applies to the forwarder role.
- *Key management* handles all secrets required for confidentiality and authenticity and thus applies to the publisher and subscriber role.

ROLE CONSIDERATIONS The concept of roles in Pub/Sub has been introduced in Section 3.1 (page 35). The three roles publisher, subscriber, and forwarder (also called broker) allow for a split of Pub/Sub nodes according to these roles

Figure 13 depicts an example that splits each participant into the three role-based building blocks publisher \mathcal{P} , subscriber \mathcal{S} , and forwarder \mathcal{F} . The left most participant acts as mere publisher and thus

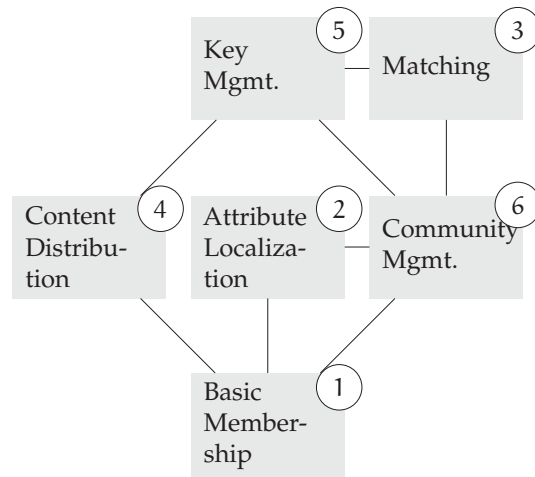


Figure 12: Design considerations—building blocks and interactions. The numbers indicate the lifecycle phases.

does not require the subscriber building block. The bottom left participant only acts as forwarder, the bottom center participant as forwarder and subscriber, and the right most participant assumes all roles. The top center participant is not active in any role.

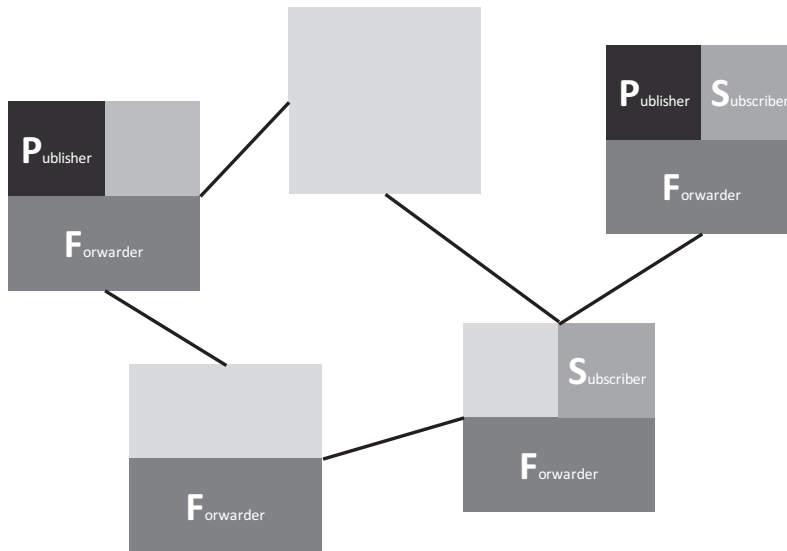


Figure 13: Building blocks—role considerations

Depending upon the role, some of the building blocks do not have to be considered, i.e., such building blocks can be disabled. For instance, a node not taking the publisher or subscriber roles is not required to provide the key management building block.

LIFECYCLE CONSIDERATIONS Beside roles, runtime phases of a system constitute another method to split building blocks. The following phases reoccur in *Pub/Sub* systems (see Figure 12):

1. First, a participant has to connect to the system. This phase is covered by *basic membership*.

2. Next, the participant locates and subscribes to attributes—*attribute* localization.
3. To transport notifications to the participant, forwarders have to decide if a notification matches a subscription via the *matching* building block.
4. Furthermore, notifications have to be distributed with minimal overhead and confidentiality, which is covered by the *content distribution* building block.
5. To protect confidentiality, cryptographic keys are required that are managed using the *key management* building block.
6. Finally, connected nodes form a community that is subject to churn and node failure, which have to be compensated via the *community management* building block.

Depending upon the lifecycle phase, a node does not need to provide all building blocks. For instance, the matching and attribute localization building blocks are not required while a node is distributing content.

3.6 BUILDING BLOCKS

Having split anonymous Pub/Sub systems into building blocks according to Figure 12, this section discusses existing solutions to realize each building block.

3.6.1 Basic membership

To goal of basic membership is to supply every participant with neighbors to connect with and to maintain these connections. Following the formalization, basic membership management manages $N(v)$ for every node v . Other building blocks then use these connections (Figure 12).

In detail, basic membership performs the following tasks: First, basic membership needs to supply every participant with an initial neighborhood set. This enables a participant joining the anonymous Pub/Sub system for the first time to connect. That means, Constraint (29) must be fulfilled as all times. Without this constraint, nodes of the basic overlay G could be disconnected and not be able to fulfill the publication and subscription integrity requirements.

$$\forall v \in V : |N(v)| > 0 \quad (29)$$

$$\forall v \in V : l \leq |N(v)| \leq u \quad (30)$$

Second, basic membership needs to maintain a neighborhood set under churn. For instance, given that participants join and leave the system, a neighborhood size within a certain bound is maintained. That includes the discovery of new neighbors, as well as the removal

of excess neighbors. Formally, ensure Constraint (30) for some defined lower boundary l and upper boundary u . The lower boundary prevents G from degenerating to a chain of nodes (SPoFs); the upper boundary prevents G from degenerating to a fully meshed graph. Every subscriber is directly connected to every publisher in a fully meshed graph, and malicious insiders can therefore trivially de-anonymize participants.

Third, basic membership needs to maintain the integrity of the connections with the neighbors to ensure no messages are lost (undetected).

Formally, basic membership supplies each participant, referred to as node v in a graph $G := (V, E)$ with $v \in V$, with a set of neighbors, referred as $N(v)$ (see formal model in Section 3.1). Hence, each node runs a membership management protocol that establishes the set of edges E of G . The edges E are assumed to be unidirectional.

Example 11. A centralized Pub/Sub system with one dedicated forwarder (broker) that runs on top of IP uses a basic overlay that differs from the underlay. The basic overlay G consists of all publishers, subscribers, and the forwarder as V . Every node shares exactly one edge with the forwarder. The underlay, however, may consist of additional nodes, e.g., proxy servers, VPN gateways, and routers.

Basic membership follows one of the underlying structures of the message dissemination approaches discussed in Section 2 (page 10). These structures (topologies) are centralized, semi-centralized, and P2P. With a centralized topology, all publishers and subscribers connect with one central forwarder. With semi-centralized, connections to multiple forwarders are possible; the forwarders are connected via either of the topologies themselves. The underlay may influence the possible basic membership topologies, e.g., friend-to-friend (F2F) networks prevent a centralized topology. The following paragraphs discuss centralized, semi-centralized and P2P topologies.

CENTRAL Central basic membership constitutes the most common Pub/Sub topology [51, 103, 118, 146]. While this topology does not provide any anonymity—except when used in conjunction with an anonymization service such as Tor as shown in [118], this topology serves as a reference for the lowest signaling overhead. Every node connects to one central forwarder. This forwarder is the only node that takes the forwarder role and is, therefore, the only node with a matching building block.

Formally, the forwarder is denoted as v_1 and every node $v_2, \dots, v_{|V|}$ has exactly one neighbor as defined by Constraint (31).

$$\forall v_i | i \in [2, |V|] : N(v_i) = \{v_1\} \quad (31)$$

$$|E| = |V| - 1 \quad (32)$$

As a result, the number of edges is minimal as shown in Equation (32). Likewise, the path length and number of messages to distribute a notification exceed optimum by only 1. However, v_1 is the

bottleneck (SPoF) for scalability as it takes over the building blocks of matching, community management, and content distribution for all other nodes. Furthermore, this node obtains global knowledge of all messages (attacker model G2), and can thus break subscriber and publisher anonymity, unlinkability, and relationship anonymity.

SEMI-CENTRALIZED A semi-centralized topology mitigates the SPoF drawback via additional forwarders. Typically, every node connects to one forwarder, and forwarders interconnect via another basic membership approach.

Formally, k nodes v_1, \dots, v_k take the role of forwarders for nodes $v_{k+1}, \dots, v_{|V|}$. As a result, the $|V| - k$ nodes only require one edge to connect to one of the k nodes. Within a centralized topology v_1 ($k = 1$) is the only forwarder; all $v_i | i \in [2, k]$ connect to v_1 . The number of edges remains unchanged as shown in Equation (33).

$$|E| = (|V| - k) + (k - 1) = |V| - 1 \quad (33)$$

For this approach, failure of v_1 segments the system (SPoF). This node also monitors the communication with all other nodes (attacker model G2). As this attacker is also internal, i.e., can read all messages, it can deanonymize all participants. Meshed connectivity between forwarders may prevent segmentation. Regarding overhead induced by this approach, the space required by each of the k forwarders to manage subscribers and publishers requirements for the other forwarders by approximately $(k - 1)/|V|$. Hence, the scalability improves even with a centralized forwarder basic membership significantly. The central forwarder does not learn about the messages of normal participants compared to the central approach. However, the collusion of the k forwarders still allows breaking all anonymity requirements as with the centralized topology.

FULLY DISTRIBUTED (P2P) A P2P topology interconnects *peers* directly without any forwarders. A static neighborhood definition as with the previous two topologies, i.e., the forwarders are predefined for all publishers and subscribers, does not work for P2P. Every peer can take the forwarder role, and peers are subject to churn (join and leave the system). To overcome this issue, P2P systems provide neighborhood discovery mechanisms such as gossiping [61]. Also, P2P systems may enforce a structure among the nodes (cf. Section 2.4, page 27). Both, structured and unstructured Pub/Sub systems exist. Scribe [125] uses a DHT (structured) for topic-based Pub/Sub. Flooding can be used as advertisement approach in unstructured P2P [137].

Structured overlays provide better guarantees, e.g., the number of messages and path lengths, than unstructured overlays. However, a structure, e.g., DHT can also be abused by malicious insiders: such an insider could obtain an ID that is responsible for a certain attribute a . Having such an ID, this attacker could observe messages related to this attribute. With a structure, an attacker can, therefore, obtain

PROPERTY	CENTRAL	SEMI-C.	P2P	
			STRUCT. P2P	UNSTRUCT.
Signalling overhead	+	o	o	o
Space overhead	-	-	+	?
Path length	+	o	o	o
Failure resilience	-	o	+	+
Anonymity	-	o	o	+

Table 2: Comparison of basic membership approaches. The symbols +, o, - indicate how well the approaches perform for the given properties.

a “position” similar to a forwarder in a centralized topology. Protocols for unstructured P2P systems create randomized overlays. Within such an overlay, the attacker cannot obtain an ID as with structured overlay to monitor message. However, unstructured systems cause more signaling overhead for the attribute localization component as messages are handled by multiple peers before reaching the destination.

Alternatively, a P2P-based Pub/Sub basic membership can also be “piggybacked” on top of existing overlays like Darknets [24] or OSNs like Twitter [35]. The basic membership management only needs to provide the capability to communicate with neighboring nodes, independent of the depths of the underlying communication stack.

SUMMARY Table 2 summarizes the discussion. The table is structured according to the properties discussed in this section: the signaling overhead *Msgs.*, the storage overhead *Space*, short transmission paths *Paths*, failure tolerance / not SPoF *Resilience*, and how suitable the approach is to protect *Anonymity*.

Centralized topologies cause the lowest signaling overhead, while P2P is favored for anonymous Pub/Sub [103, 154]. Structured P2P mitigates the drawbacks in space overhead and resilience at the cost of more message overhead and longer paths. Semi-centralized approaches perform in between centralized and structured P2P. Compared, unstructured overlays induce more space and message overhead, but maintain better resilience against failures. Furthermore, unstructured overlays provide better anonymity as no single point of observation can be determined.

3.6.2 Attribute localization

The attribute localization building block is responsible for establishing paths between publishers and subscribers, i.e., localize which nodes use a given attribute. Attribute localization tells publishers

where to send messages to, given an attribute; attribute localization tells subscribers where to subscribe to for a certain attribute.

Attribute localization has to fulfill the following tasks: First, it has to ensure that the attribute localization is complete, i.e., that a path to every publisher and every subscriber for an attribute is established (requirement subscription integrity). Second, attribute localization must complete with subscription confidentiality. The attribute of subscriptions and advertisements should remain on the attackers global observer and malicious insider; assuming the malicious insider is not a publisher or subscriber for the respective attribute. Third, attribute localization should maintain a low overhead concerning signaling and space.

Attribute localization can either follow a proactive approach, i.e., attribute knowledge is distributed before a subscriber attempts to localize, or a reactive approach, i.e., subscribers have to localize attribute knowledge by themselves upon interest. Proactive attribute localization can be also referred to as *Pub/Sub with advertisements* [51].

Proactive attribute localization influences the signaling overhead. It may even distribute attribute knowledge for attributes that will be never subscribed to. Likewise, reactive attribute localization can cause higher signaling overhead for popular attributes. Hence, proactive best fits popular attributes whereas reactive best fits unpopular attributes.

Attribute localization highly depends upon basic membership; a similar structure is used here. This thesis distinguishes centralized, decentralized, as well as *P2P* approaches. Either approach can be performed proactively and reactively.

CENTRAL FORWARDER A central forwarder maintains all knowledge for attribute localization in one place. All nodes place their subscription at this single forwarder. Consequently, all notifications are distributed via this single forwarder.

This method provides complete localization. Furthermore, like for basic membership management, the message overhead is low ($|\mathcal{P}_a| + |\mathcal{S}_a|$ messages per attribute $a \in \mathcal{A}$). Likewise, the space overhead remains low. However, the central forwarder also becomes a *SPoF*. Like with the basic membership building block, the central forwarder represents the global observer threat $\mathcal{G}\mathcal{O}$ which can, therefore, observe all messages. As a consequence, this attribute localization cannot provide anonymity.

DECENTRALIZED To overcome the *SPoF* issue of a central forwarder on anonymity and availability, the functionality can be distributed among some or all nodes. For a semi-centralized attribute localization, the same arguments as for basic membership apply. For a fully distributed approach, the following methods for implementing a localization building block can be considered: flooding, random walks, and *DHTs*.

FLOODING With flooding, knowledge is distributed to all nodes from neighbor to neighbor. For instance, with *blind flooding* [170] knowledge is distributed to every neighbor except the incoming one, similar to the breadth-first search (BFS) algorithm. This mechanism is used by several P2P applications, e.g., Freenet [24], Gnutella [157] (to all nodes, a TTL terminates) and Kazaa (only among super peers).

Flooding minimizes the delivery time of notifications as subscribers can subscribe via the path with the shortest transmission time. However, the proactive distribution of attribute knowledge requires up to $|E|$ message per attribute. Hence, a centralized approach with Equation (34)—holds for all basic membership topologies except a chain of nodes—causes less message overhead. Furthermore, this approach requires $|N(v)|$ space for every node per attribute, less than the $|V|$ entries for a central broker, but replicated for every node.

$$\overbrace{|\mathcal{P}_a| + |\mathcal{S}_a|}^{\text{central}} \leq |V| \leq \overbrace{|E|}^{\text{P2P}} \quad (34)$$

RANDOM WALK The random walk is an algorithm used for graph traversal, for instance to lookup information in unstructured P2P overlays (cf. Section 2.4.2, page 28). Random walks are probabilistic by nature regarding the traversed path [96]. Every message is forwarded via a randomized path through the basic overlay. In case the message loops or no more neighbor is available, the random walk has to backtrack or restart.

Given the probabilistic nature of random walks, the algorithm typically uses two termination criteria, the successful traversal, and the upper limit on the path length or execution time. As a result, a random walk may not succeed in traversing a graph, e.g., locating an attribute. The probability of a successful random walk is given in Equation (35) according to [96] where p denotes the average probability of finding the attribute at a node, and H the number of hops the walk performs, which is thus the message overhead H . Applying this formalization to the previously presented Pub/Sub system formalization, the probability p corresponds to the density of publisher among all nodes. This density can be defined as Equation (37) given an attribute a . For low densities, the message overhead H is high. Multiple (k) simultaneous random walks [13] only reduce localization time but not message overhead: for k random walks the success probability becomes Equation (36) according to [96] and, therefore, the message overhead kH .

$$P(p, H) = 1 - (1 - p)^H \quad (35)$$

$$P(p, H, k) = 1 - (1 - p)^{kH} \quad (36)$$

$$p_a = |\mathcal{P}_a| \div |V| \quad (37)$$

DHT In case the basic membership management uses a structure, this structure can be used for attribute localization. For instance, a

PROPERTY	CENTRAL	P2P		
		FLOOD	RANDOM W.	DHT
Completeness	+	+	-	+
Confidentiality	-	+	+	-
Signalling overhead	+	-	o	o

Table 3: Comparison attribute localization approaches

DHT approach such as Chord [30] maps both, node **IDs** and attribute, to the address space of a **DHT** via a hash function. The node with the hash value closest to the hash value of the attribute serves as the forwarder for this attribute (rendezvous point).

The characteristic of the hash function, as well as, the use of so-called finger tables lead to $\log|V|$ messages to find an attribute. Likewise, Chord only requires $\log|V|$ space per attribute per node. However, this approach requires global node **IDs**. Therefore, an attacker may distinguish nodes, and thus senders and recipients of a message with ease, and link them as well. Moreover, an internal attacker could exploit knowledge of the hash of an attribute to obtain the closest node **ID** and take control as designated forwarder. Hence, the attacker would learn \mathcal{S}_a and \mathcal{P}_a immediately.

SUMMARY Table 3 summarizes the discussion. Centralized approaches benefit from low message overhead and can easily guarantee complete localization. However, subscription confidentiality cannot be preserved. Structured **P2P** also guarantees complete localization with reasonable overhead. However, global node **IDs** allow to link message and thus potentially violate subscription confidentiality. Flooding as unstructured approach provide localization completeness as well, does not depend on global node **IDs** but induces high message overhead. Probabilistic approaches such as random walks do not require global **IDs** as well, have less message overhead, but cannot guarantee complete localization.

3.6.3 Key management

Cryptography is required to provide confidentiality, authenticity, and anonymity as well. Therefore, keys have to be managed: generated, stored, exchanged, renewed, and destroyed/revoked. While cryptographic primitives are out-of-scope of this paper, key management is a critical building block of anonymous **Pub/Sub** as key management may leak information and thus break anonymity.

Example 12. *Asymmetric cryptography via the RSA [123] scheme is commonly used to sign messages digitally. While a RSA key pair is not necessarily bound to a node **ID**, RSA key pairs constitute their unique **ID** in combination with a certificate. Therefore authentic (digitally signed) messages are linkable, even if the basic membership protects anonymity.*

Besides key IDs, the distribution of keys between nodes must be realized in an anonymity-friendly manner. For instance, subscribers \mathcal{S}_a must be able to obtain decryption keys for messages encrypted by publishers \mathcal{P}_a while preserving unlinkability.

In detail, key management must fulfill the following requirements:

- **Anonymity.** Key escrow protocols must protect the anonymity of participants. In particular, publisher and subscribers cannot be linked together by an attacker.
- **Offline capabilities.** Content distribution (notifications) should work offline, i.e., without requiring online key exchange or verification. Online activities would allow an attacker to link notifications with participants.
- **Revocation / re-keying.** The key escrow should support dynamics of a Pub/Sub system such as churn within \mathcal{S}_a and \mathcal{P}_a by revoking keys.

An anonymous Pub/Sub system must provide confidentiality and authenticity between publishers and subscribers as well as between neighboring peers. Otherwise, the global observer could infer message relations, and the malicious insider could forge and replay messages. Cryptographic keys and certificates are required to provide confidentiality and authenticity of advertisements, subscriptions, and notifications.

The following alternatives can be distinguished: out-of-band key exchange, a TTP, and decentralized key exchange.

OUT-OF-BAND An out-of-band key exchange, which is not performed via the system under review, is often used for privacy-preserving communication systems [162] as it circumvents the challenge of information disclosure. Methods are manifold: keys can be posted on websites, sent via e-mail, exchanged within close vicinity via NFC, ad-hoc networks, and via QR-codes. The latter mechanism bridges the gap between the digital and physical world as keys can be even exchanged in printed form, e.g., via a postal service.

An out-of-band key exchange does not leak any information to the global observer threat $\mathcal{G}\mathcal{O}$. The malicious insider $\mathcal{I}\mathcal{A}$ may break anonymity within an out-of-band key exchange. In particular, direct key exchange between participants, e.g., via mail, cannot ensure relationship anonymity. To analyze anonymity correctly, additional attacker models should be considered for out-of-band key exchange. Key revocation can be considered expensive out-of-band as either a revocation list or re-keying has to be performed.

TTP-BASED A TTP can disseminate keys in different variations: a single TTP that knows all keys, a separate TTP for attribute localization and content distribution keys each [85], or two TTPs that separate authorization and key escrow. For the latter option, a participant first connects to the first TTP, obtains an authorization token, then

PROPERTY	OOB	TTP	DECENTR.
Offline capability	-	+	+
Anonymity	o	-	o
Revocation	-	+	+

Table 4: Comparison key exchange approaches.

anonymously connects to the second TTP, presents the authorization token, and obtains the key. A TTP can be also used in offline and online mode. For the offline mode, the TTP issues keys combined with certificates with a limited time validity. During the validity period, participants can use the keys and validate them via the certificates without contacting the TTP.

Certain TTP functions can be performed offline. However, the TTP is still a SPoF. Therefore, a TTP is an attractive target to break availability, confidentiality, authenticity, and potentially anonymity as well. Furthermore, the TTPs must remain decoupled from centralized forwarders to ensure no single entity possess keys and can observe messages simultaneously, i.e., becomes a very powerful malicious insider.

DECENTRALIZED Decentralized key escrow approaches overcome the SPoF drawback of TTPs. Such approaches split participants into subgroups, each subgroup managed by a separate TTP, and these TTPs connect hierarchically [117]. Key distribution in such topologies can then, for instance, be realized via multicast protocols.

SUMMARY Table 4 summarizes the discussion. The out-of-band key exchange is offline, but does not provide suitable revocation and may not provide anonymity. Compared, a TTP can provide offline mode and revocation, but can hardly protect anonymity without responsibility split among several TTP entities. Decentralized approaches improve over a single TTP concerning anonymity due to decoupling.

3.6.4 Matching

The building block matching is responsible for routing decisions of notifications. That is, deciding given a subscription and a notification if the notification should be forwarded to the subscriber or not. Matching is crucial for confidentiality as notification and subscription confidentiality must be persevered when a third party, e.g., a forwarder, matches them.

Formally, matching can be decomposed into the two functions *match* and *cover* [51] (cf. Section 2.1.2, page 13) and represented in the model used in this thesis:

MATCH Given a notification m_{notif} and a subscription m_{sub} , *match* is defined by Function 38.

COVER The cover function merges subscriptions to reduce space overhead. Given two subscriptions, $m_{sub,1}$ and $m_{sub,2}$, cover is defined by Function 39. It thus returns true if $m_{sub,1}$ is more general and thus covers $m_{sub,2}$.

$$match(m_{notify}, m_{sub}) \begin{cases} \text{true :} & \mathcal{A}(m_{sub}) \subseteq \mathcal{A}(m_{notif}) \\ \text{false :} & \text{otherwise} \end{cases} \quad (38)$$

$$cover(m_{sub,1}, m_{sub,2}) \begin{cases} \text{true :} & \mathcal{A}(m_{sub,1}) \subseteq \mathcal{A}(m_{sub,2}) \\ \text{false :} & \text{otherwise} \end{cases} \quad (39)$$

Unlinkability as part of anonymity is a major challenge with matching: the output of the match function exactly determines if a subscription links to a notification or not. Hence, unlinkability cannot be fully preserved with matching. However, linkability between other message combinations, e.g., two subscriptions, can be prevented.

In detail, matching must comply to the following requirements:

- **Confidentiality.** The functions match and cover must not leak the contents of notifications and subscription, in particular not the attribute, towards the executing node.
- **Anonymity.** Matching and cover must not leak IDs of publisher and subscriber. Furthermore, linkability of messages should be limited as much as possible while it cannot be prevented.
- **Low computational overhead.** Matching has to be performed whenever a routing decision is necessary. Hence, computational overhead must be minimized.

The following alternatives can be distinguished: pseudonyms, Bloom filters, private matching, order-preserving cryptography, and zero-knowledge proofs (ZKPs).

PSEUDONYMS A pseudonym denotes a pseudonymous ID, a replacement for a real name or ID. A pseudonym is a mapping from the space of IDs or attributes to another space, the pseudonym space [97]. The mapping has to be designed such that the pseudonym cannot be linked to the attribute. Therefore, a secret is required to establish this mapping, e.g., via a keyed hash function. Pseudonyms can be associated with subjects and information objects [113]. Pseudonyms are used in Pub/Sub to protect confidentiality [146] as well as to locate information in P2P systems [100].

Pseudonyms can be categorized along many dimensions [112], for instance to what a pseudonym refers to. A *person pseudonym* is used by only one person; a *group pseudonym* can be used by multiple persons. Such a group pseudonym can be related to a particular role, a *role pseudonym*. A pseudonym can also refer to a specific type of use. A *relationship pseudonym* is only used within the context of one particular relationship between persons; the type of relationship, as well as, the person define the relationship pseudonym. Pseudonyms

categories can be also combined to form new types of pseudonyms, such as the *role-relationship pseudonym*. A pseudonym may also refer to just one single action—the *transaction pseudonym*. The pseudonym types differ in how linkable the performed actions, e.g., sent messages, are to the associated person(s). A transaction pseudonym is most unlinkable, whereas the person pseudonym is most linkable.

Pseudonyms are computationally efficient to create, e.g., via a keyed-hash message authentication code (HMAC) function, and are also efficient to compare. Furthermore, pseudonyms protect confidentiality assuming it is first computationally hard to reverse them, and it is hard to create a dictionary without possession of the secrets.

BLOOM FILTERS Bloom filters [15] are probabilistic data structures that allow testing if an element is likely to be contained in the structure. Bloom filters fulfill the following two properties:

1. An element that is part of the Bloom filter is guaranteed to test positive.
2. A positive test does not guarantee the element is indeed in the Bloom filter.

Due to the second property, Bloom filters are well suited to protect confidentiality. Bloom filters may produce false positives (FPs) but no false negatives (FNs) and, therefore, ensure functional correctness in the sense that no notifications are missed despite the probabilistic nature of the filter.

Formally, a Bloom filter \vec{bf} is a m -bit binary array, and a set of k hash functions H . For an input element a , a Bloom filter can be defined as Function (40). Each hash function sets exactly one of the m bits to 1 or sets nothing at all. Hence, $k \geq m$ hash functions are required. Every \vec{bf} is initialized with all m bits set to 0.

$$\vec{bf} = \forall i \in [1, m] : \vec{bf}[i] = \max_{j=1}^m H_j(a) \quad (40)$$

$$\forall i \in [1, m] : \vec{bf}[i] \geq \vec{bf}'[i] \quad (41)$$

To create a Bloom filter, every hash function is applied to the input elements and may set the corresponding bit to 1. A bit that is already 1 will remain 1 independent of additional elements added to the filter. To test if an element a is contained in a Bloom filter \vec{bf} , one creates \vec{bf}' from a , and test for every bit in \vec{bf}' if this bit is 1, and if so, the corresponding bit in \vec{bf} is 1 as well as shown in Equation (41). Note that 0's in a filter are not considered for testing.

A Bloom filter can also hold more than one element, i.e., it supports the cover function. For that, additional elements are consecutively added to the Bloom filter. However, additional elements increase the probability of FPs [15]. This drawback of FPs for Bloom filters can be partially mitigated via counting Bloom filters [53], which were designed to allow for element deletion from Bloom filters. A counting

REQUIREMENT	PSEUDO.	BLOOM F.	PEKS
Confidentiality	o	o	+
Anonymity	o	o	o
Minimal overhead	+	+	-

Table 5: Comparison of matching approaches.

Bloom filter consists of m buckets of n bits that count the 1's. Therefore, the minimal count of 1's per position can be compared, which gives a lower FP rate.

Bloom filters are efficient to create and to compare. Furthermore, they are space efficient for multiple attributes. Bloom filters can be used to prevent linkability by adding noise, i.e., setting random bits to 1. Hence, two identical subscriptions could use different Bloom filters. However, to protect confidentiality, m must be chosen small to create more FPs and thus render brute force attack non-economical. Bloom filters are used for privacy-preserving Pub/Sub systems [9, 84, 118, 139].

PRIVATE MATCHING Private matching schemes, e.g., public key encryption with keyword search (PEKS) [16], are a derivation of asymmetric cryptography that provides an additional *match* function beside encrypt and decrypt. Match only requires two ciphertexts and no secrets, and thus can be performed by a third party. The first ciphertext has to be encrypted with the secret key whereas the second ciphertext has to be encrypted with the public key. The ciphertext created with the public key can be furthermore randomized such that two ciphertexts created with the same key cannot be linked.

Private matching is used in privacy-preserving Pub/Sub [79, 118, 136, 139] and ensures confidentiality. Furthermore, private matching can prevent likability due to ciphertext randomization. However, an attacker in possession of a publicly and a corresponding privately encrypted ciphertext can also link all additional ciphertexts. Furthermore, asymmetric cryptography is less efficient than hash functions and, therefore, imposes higher computational overhead as pseudonym and Bloom filters.

SUMMARY Table 5 summarizes the discussion. Pseudonyms and Bloom filters impose low computational overhead but have shortcomings with confidentiality and anonymity, depending the assumptions. Cryptographic approaches such as PEKS provide good confidentiality but also have shortcomings for anonymity, and cause computational overhead.

3.6.5 Community management

On the basis of the basic membership and attribute localization, publishers, subscribers, and forwarders establish attribute meshes \mathcal{M}_a

per attribute a with the support of the matching building block. These attribute meshes—or communities—have to be maintained, e.g., in the presence of node churn. Furthermore, such overlays can be optimized. Such optimization criteria can be for instance:

MINIMAL MESSAGE OVERHEAD The number of messages is minimized by removing forwarders from the attribute mesh.

MINIMAL DELAY The transmission delay between publishers and subscribers is minimal given the path lengths in the attribute mesh are minimal. This can be achieved by increasing the fanout degree.

RESILIENCE AGAINST NODE AND EDGE FAILURES Resilience against such failures can be reduced by maintaining alternative connections from the basic membership at increasing space overhead.

Possible optimizations and inter-dependencies with other optimizations goals are manifold. However, community management highly depends on the basic membership, e.g., if new edges are possible or not, and other building blocks.

3.6.6 *Content distribution*

Content distribution ensures that subscribers receive the desired content. Therefore content distribution must prevent message loss, ensure correct order, and protect the confidentiality of notifications. Many of the [Pub/Sub](#) security requirements apply to content distribution: confidentiality, integrity, authenticity, anonymity. In detail, the content distribution must realize the following functionalities.

First, the content of notifications (payload) must be transported confidential. Second, notifications must not be lost and reach all subscribers given a particular attribute (notification integrity). Third, notifications should be authentic in the sense that subscribers can verify that the notifications originated from some publisher given a particular attribute and that notifications have not been tampered with. Content distribution must not leak any information leading to a de-anonymization of subscribers or publishers (publisher and subscriber anonymity). Furthermore, notifications should not be linked to each other, e.g., expose that two notifications related to the same attribute, publisher, or subscriber.

To comply with these requirements, content distribution relies on key management. The approaches asymmetric cryptography, multi-layer encryption, and attribute-based cryptography are discussed here.

ASYMMETRIC ENCRYPTION AND DIGITAL SIGNATURES Asymmetric cryptography uses two different keys, a public key pk to encrypt, and a secret key sk to decrypt. This approach is valuable to [Pub/Sub](#) as this cryptography strongly separates the roles of encrypter and decrypter—publisher and subscriber. Therefore, a subscriber cannot take the role of a publisher and encrypt information. Moreover,

the term public key does not imply public to everyone, but can be rather shared with a set of participants, e.g., \mathcal{S}_a .

The common asymmetric encryption scheme RSA [123] works as follows:

GENERATE KEYS Select primes p, q and compute $n = pq$ as well as $n' = (p-1)(q-1)$. Select $e \in (1, n') \subset \mathbb{N}$ with $\gcd(e, n') = 1$, i.e., e is not a divisor of n' . Calculate $d = e^{-1} \bmod n$. Then the secret key is defined as $sk = (d, n)$ and the public key as $pk = (n, e)$.

ENCRYPT(m, pk) With plaintext m calculate ciphertext c via $c = m^e \bmod n$. Short notation: $c = \mathcal{E}_{pk}(m)$.

DECRYPT(c, sk) With ciphertext c calculate plaintext m via $m = c^d \bmod n$. Short notation: $m = \mathcal{D}_{sk}(c)$.

SIGN(m, sk) With plaintext m calculate a message digest $h = H(m)$ and obtain signature $\text{sig}_m = \mathcal{E}_{sk}(h)$.

VERIFY(m, sig_m, pk) With plaintext m and signature sig_m check if $H(m) \stackrel{?}{=} \mathcal{D}_{pk}(\text{sig}_m)$.

Asymmetric cryptography can be used to achieve confidentiality, and in combination with a message digest integrity and authenticity as well. Asymmetric cryptography imposes low computational overhead in combination with symmetric encryption.

MULTI-LAYER ENCRYPTION Multiple layers of encryption can be used to prevent a single internal attacker from learning unnecessary path information, for instance, onion encryption [151]. With this onion encryption, a node only learns predecessor and successor in a message flow. Hence, this approach protects against an internal attacker. However, the sender must define the full path and thus knows the public keys of every node on this path, which violates anonymity.

An anonymous Pub/Sub system that builds upon unstructured P2P basic membership gains no benefits from onion encryption, as each node can only learn predecessor and successor due the lack of global IDs. As onion encryption imposes computational overhead, it is only useful for systems with basic membership management that require global node IDs.

The multiple layer commutative encryption (MLCE), e.g., used in [137] for secure Pub/Sub, does not require the sender to know the full path in advance, but rather allows to add and remove encryption layers in arbitrary order. However, MLCE still induces computational overhead and thus also only benefits basic membership with global IDs as well.

ATTRIBUTE-BASED ENCRYPTION The attribute-based encryption (ABE) extends the concept of asymmetric encryption to attributes. Rather

than having to possess one secret key to decrypt a message, ABE enforces the possession of one or more attributes to decrypt a message [128].

This concept can be realized via attribute keys. With attribute keys, every attribute is associated with a public/secret key pair; the possession of an attribute is equivalent to the possession of the secret key associated with the attribute. Possessing all secret attribute keys is necessary to decrypt a message, ABE uses some mechanisms to derive a symmetric decryption key for the message [128]. Such a key can be derived by methods, not unlike polynomial interpolation. Attribute keys are used to decrypt support points, and sufficient support points provide the polynomial, which is then used to derive the decryption key. Similar mechanisms have been used for fine-grained access control for a long time [4] (Chapter 13, Nuclear command & control).

BROADCAST ENCRYPTION Broadcast encryption [56] tackles the challenge of distributing a secret key to many receivers while being able to revoke receivers. Assuming one sender and a group of receivers, broadcast encryption allows the sender to share a secret with any subgroup with minimal signaling overhead.

Broadcast encryption assumes that every receiver is initially equipped with some keys. Then the sender splits the secret into secret shares and sends them encrypted in such a way to the group of all users that only the desired subgroup of receivers can restore the shared secret.

3.6.7 Guideline for Pub/Sub system construction

Constructing a privacy-preserving Pub/Sub systems can be challenging considering the interdependencies of the discussed approaches. This section provides guidelines for building anonymous Pub/Sub systems by discussing alternatives in the context of application requirements.

The *matching* building block influences the available options for other building blocks the most. Thus, it should be considered first. If the performance of forwarders is most important, *pseudonyms* [97, 100, 113, 146] and *Bloom filters* [9, 15, 84, 118, 139] are good options. If composite subscriptions are required, (counting) Bloom filters [53] perform well. Attribute *key overlay trees* [17, 153] are well suited when attribute value comparison is required. Private matching requires more computational overhead but does not lead to FPs as Bloom filters do, and provides stronger unlinkability than pseudonyms. Pseudonyms, filters, and private matching are compatible with all types of *attribute localization* and *membership management*. Key-based overlay trees require end-to-end connectivity between subscribers, and participants have to establish a tree structure, which is incompatible with random walk and flooding based mechanisms. If few attributes and many subscribers are expected, proactive, e.g., via flooding, will result in the least overhead and short delivery path. In the case of hardly used

attributes, a random-walk based proactive subscription is the better option.

Regarding community management, it is beneficial regarding resilience to failures and attacks to maintain more than one route per node to other nodes of the overlay. This overhead is compensated by quicker overlay repair with lower message overhead. If publishers distribute many publications over a relatively stable overlay, optimizations after community establishment, e.g., via Bullet [87], pay off. However, community optimizations may leak information to adversaries.

Key management and content distribution both depend on matching. In case many participants are expected, shared keys or groups keys, as well as a key escrow service, are desirable. In such a scenario, re-keying performs better than revocation, but longer periods of confidentiality violation have to be acceptable.

This thesis uses the following methods throughout the next two chapters: SCAMP [61] for basic membership management as an unstructured P2P network that provides neighborhood sets and end-to-end connectivity. Attribute knowledge is distributed proactively via flooding. Pseudonyms are used for matching. An offline TTP as the basic method for key management as to keep the focus on anonymity rather than confidentiality.

3.7 ANONYMOUS Pub/Sub SYSTEMS

This section discusses related anonymous Pub/Sub systems on the formal system model and the building blocks. First, a short overview of the development of anonymous Pub/Sub is provided. Second, selected anonymous Pub/Sub systems are discussed in detail.

3.7.1 Development of anonymous Pub/Sub

Wang et al. [162] first introduced anonymity as a requirement for secure Pub/Sub. In particular, they pointed out that Pub/Sub may provide anonymity against the passive malicious insider, a weaker model of $\mathcal{I}\mathcal{A}$, assuming each path between publisher and subscriber contains, at least, three nodes in the basic overlay. This assumption follows the proxy server principle (cf. Section 2.2.1, page 14). However, a detailed solution was not provided. Key management, a building block not only relevant for anonymity but also confidentiality, has also been left as an open challenge.

Datta et al. [34] proposed the first anonymous Pub/Sub system. They applied topic-based Pub/Sub to P2P networks. Every topic is represented by a dedicated P2P overlay. One special basic overlay contains super peers, who know some nodes from the topic overlays. To join a topic overlay, a new node must flood a request in the basic overlay to be directed to a node from the desired topic overlay. The paper mostly tackles the challenges of ensuring that the edges of the over-

lays are directed such that no cycles occur, and all nodes are reached. Anonymity is not analyzed against common attacker models.

The key management challenge was addressed by Srivatsa and Lui [146] as well as Raiciu and Rosenblum [118] via an online [TTP](#). The [TTP](#) issues cryptographic keys to ensure confidentiality and authenticity. In the case of Srivatsa and Lui, the [TTP](#) also issues pseudonyms to participants to provide publisher and subscriber anonymity within the limits of pseudonyms. This approach protects anonymity against the [I \$\mathcal{A}\$](#) , assuming that the [TTP](#) is not compromised. Both systems [118, 146] still rely on a central forwarder and thus assume that this node is not compromised. Also, no anonymity protection against [G \$\mathcal{A}\$](#) is possible. Chen et al. [79] distributed the forwarder role. In addition they used the searchable ciphertext method [PEKS](#) for matching to protect confidentiality. However, the key management remains out-of-band.

Parallel to searchable ciphertext, also first [P2P](#)-based approaches surfaced, e.g., Shikfa et al. [136, 138] and Tariq et al. [153]. These approaches explicitly address anonymity and also include the key management building block. However, the anonymity attacker model remains limited resulting in leaked information that can be used by a stronger attacker to break publisher and subscriber anonymity. Shikfa et al. require nodes to obtain some information about the path towards the respective publisher or subscriber. A malicious insider can use this information to reduce anonymity sets. The approach from Tariq et al. allows subscribers to learn about other subscribers with the same or similar attributes. Likewise, the malicious insider can use this information to adjust anonymity sets.

Nabeel et al. [102, 103] added homomorphic cryptography to allow matching for content-based [Pub/Sub](#), i.e., compare values of key/-value pairs, while preserving confidentiality. Di Crescenzo et al. [27] added oblivious transfers ([OTs](#)) to prevent a central [TTP](#) to learn the attributes of publishers and subscribers. This [TTP](#) remains a [SPoF](#) for availability, but it cannot break anonymity as a malicious insider. The global observer may however still de-anonymize participants in these systems.

Vo and Bellovin [161] discussed anonymous [Pub/Sub](#). Besides the proposal to use anonymization services to connect to a central broker, they also proposed structured [P2P](#) networks. Assuming that nodes can use some anonymous addressing, they can interconnect via a [DHT](#). In both cases, centralized and structured [P2P](#), Bloom filters are used as subscriptions to protect the privacy. The system protects the anonymity against malicious insiders. However, evaluation regarding outsiders—potentially a global observer—is missing.

3.7.2 Discussion of anonymous [Pub/Sub](#) systems

EVENTGUARD EventGuard is a secure [Pub/Sub](#) system to protect each type of operation with the system, e.g., subscribe, with a so-called *guard* [146]. The architecture of EventGuard is depicted in Figure 14. The guards are well represented by the previously in-

roduced building blocks—attribute localization for the subscription guard and advertisement guard, community management for the unsubscription and unadvertisement guard, community management for the publish guard and content distribution for the routing guard.

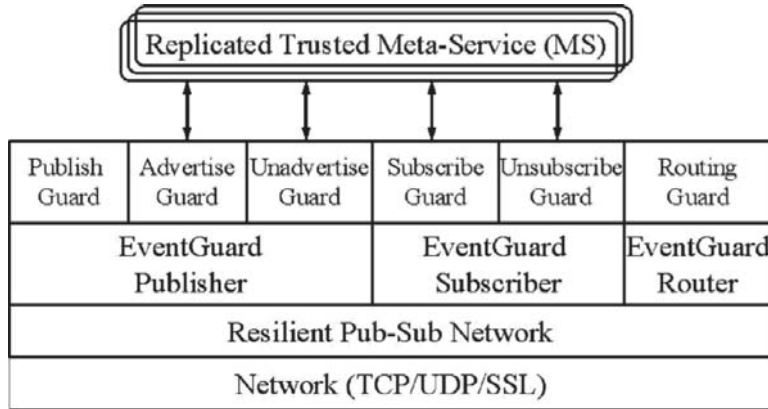


Figure 14: EventGuard architecture [146]

Within the guards, EventGuard uses tokens, keys, and signatures as mechanisms. Tokens represent pseudonyms for attributes that are being used for advertisements, subscriptions, and notifications. The keys are managed by a central and replicated [TTP](#), the *Replicated Trusted Meta-Service*. Every operation with the [Pub/Sub](#) system requires an interaction with this meta-service. ElGamal signatures [60] are used to ensure the authenticity of all messages. ElGamal signatures are probabilistic with similar properties to [PEKS](#); one message translates to many possible signatures, but every signature is only valid for one message. As the basic membership, EventGuard already assumes a fully functional [Pub/Sub](#) system.

While EventGuard does not aim to provide anonymity, pseudonyms and probabilistic signatures constitute valuable steps towards anonymous [Pub/Sub](#). The pseudonyms can be used to hide attributes during routing. The probabilistic signatures can be used to provide authenticity while preventing multiple messages to be linked to the same publishers.

C-CBPS Raiciu and Rosenblum proposed a confidential content-based publish/subscribe ([C-CBPS](#)) [118]. Like EventGuard, this system does not aim to protect anonymity by itself. However, the [C-CBPS](#) uses [PEKS](#) as encryption scheme to protect confidentiality. [PEKS](#) is a probabilistic asymmetric encryption scheme. The trapdoor produced by the receiver via the secret key is probabilistic; the searchable encryption produced by sender via the public key however can be randomized. With that mechanism, one plaintext attribute maps to many searchable encryptions, while a searchable encryption only matches exactly one trapdoor (attribute).

[PEKS](#) provides unlinkability between messages: an advertisement, containing the searchable encryption, cannot be linked to another advertisement for the same attribute. However, advertisements can be still linked together with subscriptions, i.e., the information that the

match function must leak. Having a matching subscription, advertisements may however still be linked together transitionally. To protect anonymity, the authors recommend either [DC-nets](#) or [Tor](#) for publishers and subscribers to connect to forwarders.

BROKER-LESS PUB/SUB Tariq et al. proposed a secure [Pub/Sub](#) system [154] with a [P2P](#) basic overlay and a [P2P](#) key management as well. The key management uses a variation of identity-based encryption ([IBE](#)) to ensure confidentiality. The [IBE](#)-based [P2P](#) key management also realizes attribute-localization and content distribution—all three building blocks are highly interconnected here.

The core assumption in this approach is that attributes, in this case, key/value pairs can be represented in a *trie*, an information retrieval tree. Within such a tree, the root node represents the full value range, successive child nodes represent broad value ranges, and leaf nodes represent specific values. Tariq et al. use this trie to construct the identity part for [IBE](#) keys. The keys are organized in such a way that each key of a leaf node represents the value of this node, and parent nodes inherit all child keys. As a result, the root node has the key to decrypt all values, i.e., match the whole value range, whereas leaf nodes can only decrypt one specific value.

Given this key structure, a [P2P](#) overlay tree is established where nodes take the position depending on which key they possess. Parent nodes, therefore, take the key management responsibility for their child nodes. Every notification for a specific attribute is sent to the root node of that key/value trie. This root node then recursively forwards the notification in the subtree with matching value. The tree, therefore, also handles the content distribution.

The [P2P](#) trees ensure nodes only receive messages they can decrypt, i.e., for which they subscribed. This hinders malicious insiders from gaining knowledge beyond the limits of their key material. Still, while the actions of malicious insiders are limited, subscriber anonymity is not provided.

EPIDEMIC FORWARDING Shikfa et al. proposed a so-called epidemic forwarding scheme [138]. Their approach is not a [Pub/Sub](#) system per se, but the message dissemination resembles high similarity with [Pub/Sub](#).

A distributed offline [TTP](#), [IBE](#), and [PEKS](#) are used. However, epidemic forwarding violates minimal overhead and scalability requirements, as false positives might be forwarded, and forwarders perform expensive operations.

SCRIBE Rowstron et al. proposed [SCRIBE](#) [126], a [Pub/Sub](#) system based on the [P2P](#) Pastry [125], developed by the same authors. [SCRIBE](#) does not aim to protect anonymity; however approaches of [SCRIBE](#) are well suited for anonymity.

Pastry is a structured [P2P](#) system. [SCRIBE](#), therefore, uses a structured [P2P](#) where every [Pub/Sub](#) attribute has a designated central forwarder, called rendezvous point. A publisher will always obtain the

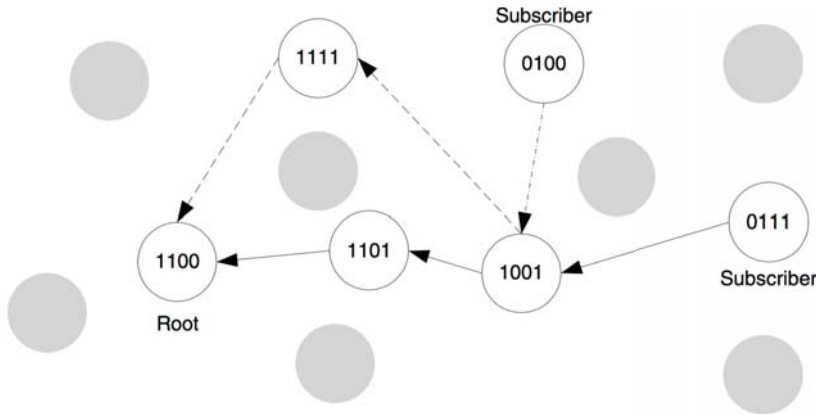


Figure 15: Base Mechanism for Subscription and Multicast Tree Creation. [126]

ID of the designed forwarder first, and send notifications directly to this forwarder. Subscribers recursively connect to the designated forwarder via their neighbor with the closest ID. These neighbors then become forwarders. The resulting network topology is a distribution tree as shown in Figure 15, routed by the chain of publisher and designated forwarder, and having only subscribers as leaves.

Assuming the designated forwarder is a malicious insider, SCRIBE cannot provide publisher anonymity. However, this malicious insider threat in the distribution tree cannot tell forwarders from subscribers. SCRIBE, therefore, may provide subscriber anonymity against a malicious insider.

3.7.3 Summary

The discussion of the related work showed that anonymous Pub/Sub progressed concerning the usage of cryptography to protect subscription privacy as well as confidentiality. This allows preserving confidentiality against forwarders (malicious insider) as well as global observers. However, the cryptography does not prevent these attackers from learning the relationship between nodes, e.g., break relationship anonymity. Furthermore, while a malicious insider is considered, the capabilities of this insider are limited: passive rather than active; only in the role of forwarders rather than publishers and subscribers. These assumptions also limit the publisher and subscriber anonymity; publisher anonymity only holds against forwarders but not subscribers (likewise for subscriber anonymity).

Table 6 summarizes this discussion with the most lacking requirements. The four anonymity requirements, publisher anonymity, subscribers anonymity, relationship anonymity, and unlinkability were each assessed by the malicious insider and the global observer (I_Q/G_Q). Key management describes if the key management is sufficiently detailed and respects the anonymity requirements.

REQUIREMENT	EVENTG.	C-CBPS	BROKER-L.	EPID.	SCRIBE
Publisher anonymity (I \mathbb{A} /G \mathbb{A})	-/-	o/-	+/-	+/-	-/-
Subscriber anonymity (I \mathbb{A} /G \mathbb{A})	-/-	o/-	-/-	+/-	+/-
Relationship anonymity (I \mathbb{A} /G \mathbb{A})	-/-	o/-	+/-	+/-	+/-
Unlinkability (I \mathbb{A} /G \mathbb{A})	o/o	o/-	-/-	-/-	-/-
Key management	+	?	+	?	-

Table 6: Comparison of related work. The symbols +, o, - indicate how well the approaches perform for the given properties. EventGuard requires add-on anonymity protection to fulfill the requirements marked as o.

3.8 SUMMARY

This section summarized the state-of-the-art in the area of anonymous Pub/Sub. For that, a formalization for Pub/Sub was created. This formalization expressed multiple network abstraction layers, node roles, attributes as Pub/Sub concept, as well as node churn. Based on this model, anonymity requirements were formulated for Pub/Sub. These anonymity requirements were complemented with an attacker model discussion. The two attacker models global observer G \mathbb{A} and malicious insider I \mathbb{A} are used in combination in this thesis as realistic attacker model. With the attacker model in mind, a novel attack for the malicious insider was presented, which abuses the request/response protocol semantic found in many network protocols and applications.

Having established a formalization and attacker model, this section proposed a segmentation of anonymous Pub/Sub in so-called building blocks: basic membership, attribute localization, key management, matching, community management, and content distribution. Dominant technologies were explained and discussed for each of the building blocks. The following chapter presents a novel anonymous Pub/Sub system by assembling the discussed technologies as well as new approaches into building blocks.

With the formalization and building blocks, the most relevant related anonymous Pub/Sub as well as comparable systems were introduced and discussed with respect to the attacker model. These sys-

tems seem to favor add-on anonymization services such as [Tor](#) and do not consider anonymity any further. As however discussed in Section 2.3.6.2 (page 23), anonymity attacks target the lack of anonymity consideration in the system. Few anonymous [Pub/Sub](#) systems do consider anonymity throughout the whole system but use a limited attacker model that greatly restricts the attackers capabilities. The next chapter, therefore, discusses anonymity for each building block with a strong attacker model.

AN ANONYMOUS UNSTRUCTURED PUBLISH/SUBSCRIBE OVERLAY

This chapter introduces a novel anonymous Pub/Sub system based upon P2P communication. The system protects publisher and subscriber anonymity against the global observer and the malicious insider threat as presented in Section 3.3 (page 41). The system is modular by design; the novel contributions to each of these modules protect against various attacker capabilities and can be selected as needed.

This chapter is structured as follows: after a brief system overview, Sections 4.1 - 4.6 introduce the system in more detail according to the building blocks as defined in the previous chapter. Section 4.3 introduces three variations for anonymous bootstrapping, i.e., the attribute localization building block, with hash chains. Section 4.5 introduces inter-overlay optimization with counting Bloom filters as a contribution to community management. Section 4.7 presents *cover traffic*, an extension of the base system that protects anonymity against malicious insiders. Section 4.8 introduces the *shell game*, an extension of the base system that protects anonymity against the global observer. Section 4.9 concludes this chapter.

The system AnonPubSub realizes anonymous topic-based Pub/Sub by utilizing an existing basic membership approach, and by contributing protocols for attribute localization, matching, and content distribution. The system enables Pub/Sub with advertisements in the sense that advertisements are distributed prior to subscriptions to aid the subscription process. AnonPubSub models key management explicitly as opposed to the out-of-band assumption found in related work. AnonPubSub also provides several means and optimizations for community management. This is the first system providing participant anonymity in Pub/Sub without relying on external anonymization services. Furthermore, the system does not assume out-of-band key management. Foremost, the system offers several means to protect anonymity against a strong anonymity attacker model and allows to balance anonymity against message overhead as needed. Several parts of this system and core contributions of this thesis are published in conference proceedings and journals [36–39].

The system is structured according to Figure 16 (right) from bottom to top.

Example 13. Figure 16 (left) provides an example for this chapter: two publishers (squares 1, 5) and two subscribers (diamonds 7, 9) form an attribute overlay such that the publishers can efficiently distribute notifications (top layer). The bottom layer shows the connections of the basic membership, i.e., the graph $G = (V, E)$. The middle layer depicts the overlay \mathcal{M} to be established via attribute localization and matching. The overlay includes forwarders (circles 3, 6), too.

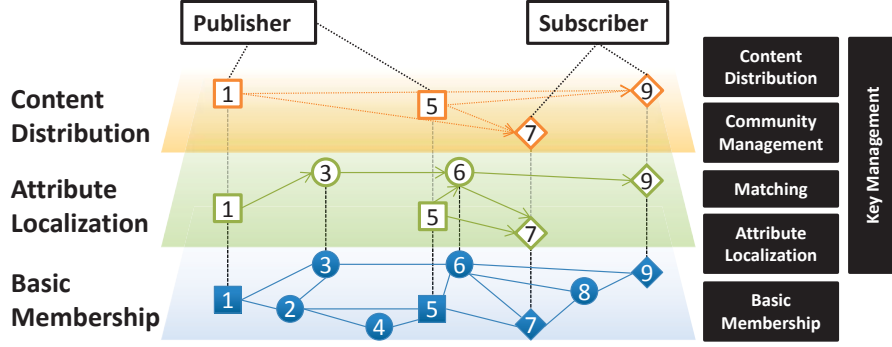


Figure 16: Basic membership (bottom), Pub/Sub overlay (middle), community (top)

The next sections are structured in accordance with the building blocks depicted in Figure 16. First, Section 4.1 introduces the basic membership via SCAMP followed by key management via a TTP in Section 4.2. Then, attribute localization in Section 4.3, and matching in Section 4.4 based upon hash chains, pseudonyms and flooding are explained. Community management in Section 4.5 and content distribution in Section 4.6 complete this chapter with a summary in Section 4.9.

4.1 BASIC MEMBERSHIP

For basic membership, we rely on the SCAMP [61] algorithm. SCAMP is a network sampling algorithm that distributes neighborhood sets (network samples) via gossiping. SCAMP ensures that every node always maintains a neighborhood set. SCAMP defines the neighborhood size for every node with Equation (42); c defines the ratio of tolerable node failures. While SCAMP provides a fixed neighborhood size, the following building blocks neither depend on a fixed nor static neighborhood. However, each connection is assumed to work both ways.

$$|N(v)| \geq (c + 1) \times \log(|V|) \quad (42)$$

Example 14. The example in Figure 16 has a variable neighborhood size $|N(v)| \in [2, 5]$ with an average of $|N(v)|_{avg} \approx 3.11$. The resulting graph diameter is 3, and the average path length ≈ 1.81 .

In addition to using SCAMP for basic membership, confidential connections are established via an initial DH key exchange. Confidential connections are crucial concerning the global observer threat to prevent the attacker from observing messages beyond the message length. The key exchange establishes a key K_{v_x, v_y} between two nodes v_x, v_y that are neighbors. Both nodes then apply symmetric encryption, advanced encryption standard (AES) in cipher-block chaining (CBC) mode to protect the confidentiality of packets transmitted via UDP. The CBC mode avoids consecutive equal messages to be encrypted to the same ciphertext. UDP prevents the exchange of un-

desired messages, e.g., flow control and acknowledgments, outside the control of the anonymous Pub/Sub system.

These properties ensure that an attacker with global observing capabilities may learn the existence and size of messages, but not the content. A malicious insider may read and generate messages. That means that the anonymous Pub/Sub system has to control the message flow to prevent inference of communication relations, i.e., protect anonymity. Furthermore, malicious insiders have to be prevented from violating the anonymity and scalability of the system, e.g., by spamming the system with false or replayed messages. The attribute localization building block covers these tasks.

To cope with node and connection failures—independent of malicious intent or not—basic membership is required to notify attribute localization about such event. Likewise, newly connected neighbors have to be reported as well. This process is further described in Section 4.3.

4.2 KEY MANAGEMENT

The system uses two offline TTPs for the initial distribution of keying material to ensure confidentiality and authenticity. The TTPs can be offline in the sense that every participant only need to connect initially to it, but functions afterwards without a connection to the TTP. Symmetric cryptography serves as basis for confidential message exchange. Asynchronous cryptography with signatures and certificates for offline verification protects the authenticity and integrity of messages. The separation of the TTP into authorization and key distribution limits the TTP from linking attributes/keys to participants.

Each participant connects with the anonymization TTP and shows one or more desired attributes as well as the desired role—publisher or subscriber. If this TTP grants the respective participant access to the desired attributes and role, it releases the corresponding keying material from the credential TTP. Publishers distributing the same attribute obtain the same key. This ensures publishers are indistinguishable from each other. Likewise, all subscribers \mathcal{S}_a interested in an attribute a are equipped with the same keying material.

TTP ARRANGEMENT A single TTP would know all participants, their attributes, and all key material. Attacking the TTP would lead to a breach of anonymity, confidentiality, and authenticity. Two separate TTPs, the anonymization TTP_{anon} and the credential TTP_{cred} , realize separation of concerns, i.e., a single TTP does not know both, node ID and attributes. This scheme ensures that TTP_{anon} relays requests from participants to the TTP_{cred} , i.e., anonymizes the participants towards TTP_{cred} , but cannot decrypt the response from TTP_{cred} .

The anonymization TTP_{anon} links node IDs and attributes of the participants, but does not know the respective keys. The TTP_{cred} links attributes and key material, but does not know participant identities. Like basic membership management, TTP_{anon} and TTP_{cred} use a

confidential channel to interconnect. This channel furthermore must provide authenticity, e.g., via transport layer security (TLS) and an out-of-band exchange of public keys.

The TTP_{cred} owns a secret key sk_{TTP}^{cred} and a public key pk_{TTP}^{cred} . Likewise, TTP_{anon} owns sk_{TTP}^{anon} and pk_{TTP}^{anon} . Both public keys are made available to all participants.

KEY GENERATION The TTP_{cred} prepares keys and certificates for all attributes with these functions: for each attribute $a \in \mathcal{A}$, it executes the function $keyGeneration(a)$ and obtains the output triple (a, sk_a, pk_a) . Furthermore, it generates a random symmetric key K_a for every attribute a as well. Hence, the TTP stores (a, sk_a, pk_a, K_a) .

For certificate generation, TTP_{cred} calculates pseudonyms t_a by executing $t_a = \{H(a)\}_{K_a}$. These pseudonyms are then used for the matching and attribute knowledge building blocks in the following two sections. Next, TTP_{cred} creates $cert_a$ by executing the function $sign(pk_a, sk_{cred})$, and stores it. In summary, TTP_{cred} maintains a tuple for each attribute a , called attribute record $record_a$ as defined in Equation (43).

$$record_a = (a, sk_a, pk_a, K_a, cert_a) \quad (43)$$

KEY DISTRIBUTION New publishers and subscribers contact TTP_{anon} to retrieve the keying material for their attributes. Publishers obtain the tuple $(sk_a, K_a, cert_a)$ for each attribute a , subscribers obtain (K_a) .

The key exchange protocol must ensure that TTP_{anon} never learns the keys, while the TTP_{cred} never learns node IDs. The keys are obtained as illustrated in Figure 17, with $a||b$ denoting the concatenation of elements a, b and $\{a\}_b$ denoting the symmetric encryption of plaintext a under key b :

1. Publisher p first generates a nonce N and a symmetric transaction key K_{tx} , and sends $(a, \{N||K_{tx}\}_{pk_{cred}})$ to TTP_{anon} .
2. TTP_{anon} verifies the eligibility of p for a , e.g., via an authentication and authorization scheme, and sends ticket $(a, \{N||K_{tx}\}_{pk_{cred}})$ to TTP_{cred} .
3. Then TTP_{cred} decrypts nonce and key, looks up key material for a , encrypts nonce and keys with K_{tx} and sends $(\{N||keys\}_{K_{tx}})$ back to TTP_{anon} .
4. Finally, TTP_{anon} sends the latter message back to p . Afterwards, the publisher decrypts the message, verifies the nonce, and obtains the keys. Likewise, subscribers obtain K_a .

This scheme ensures that TTP_{anon} never learns K_{tx} . Therefore, TTP_{anon} cannot obtain the keys and cannot link p with keys. As a result, TTP_{anon} cannot link messages, such as advertisements, to participants and attributes. Nonce N proves to p that the keys are from TTP_{cred} and TTP_{anon} behaved correctly. TTP_{cred} never communicates with p .

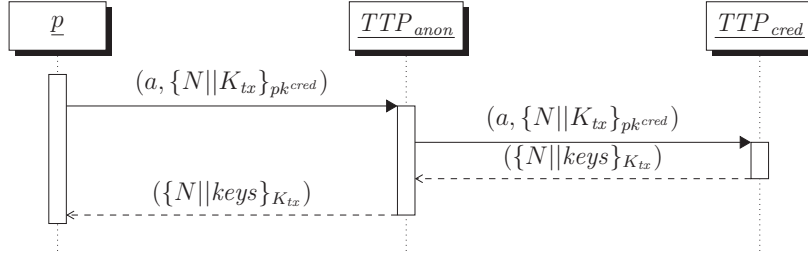


Figure 17: Key distribution sequence

Moreover, K_{tx} as well as N only leak the number of transactions per attribute to TTP_{cred} to an attacker, but not the ID of p . That means, TTP_{cred} cannot link p with a or $keys$. After that, publishers as well as subscribers do not require further TTP interaction.

4.3 ATTRIBUTE LOCALIZATION

Attribute localization serves the purpose of supplying participants where to find attributes. That is for publishers, in which direction to send notifications given an attribute; for subscribers in which direction to subscribe to attributes. Attribute localization must protect the anonymity of publishers and subscribers such that no node IDs are revealed to anonymity attackers. Furthermore, publishers and subscribers must not be linkable to attributes. That also requires that attributes themselves are kept confidential. In addition, the signaling overhead of attribute localization should be kept low. Attribute localization should also scale with the number of attributes. That means in particular that malicious insiders cannot overload the system with false attribute knowledge.

This system requires advertisements to be distributed before subscriptions are sent. That means, no subscriptions for an attribute that has not been advertised by at least one publisher before is sent. This leads to several advantages: (i) if there is no publisher for an attribute, no attribute overlay must be maintained, (ii) subscribers always learn if at least one publisher for their desired attribute is active in the system, (iii) attribute overlays can be constructed such that the hop-distance between publishers and subscribers is kept low.

AnonPubSub uses three methods for attribute localization: flooding, forest fire with random walks, and double random walks. All three methods come with respective advantages and disadvantages: flooding causes high signaling overhead for advertisements, but also creates attribute overlays with low diameter, i.e., low signaling overhead for notifications. Forest fire and the random walk reduce the signaling overhead caused by advertisements, but may lead to attribute overlays with higher diameter. While all three mechanisms are well known in the domains of computer networks and graph traversal, these mechanisms have been modified, incorporated into protocols, and extensively evaluated to allow for the construction of anonymous overlay networks. As a result, while flooding spreads

attribute knowledge to all nodes, forest fire and the random walk distributes attribute knowledge sparse among nodes. By adjusting the sparseness, the signaling overhead is balanced between advertisements and subscriptions as needed. This section first introduces attribute localization via flooding in combination with hash chains as a novel means of privacy-preserving **TTL** counter.

4.3.1 Flooding

Flooding of advertisement messages within the basic membership distributes attribute knowledge as introduced in [37]. The flooded advertisement messages contain two elements: an attribute pseudonym and a transaction pseudonym. The attribute pseudonym protects confidentiality while the transaction pseudonym helps to set up the overlay network.

A publisher $p \in \mathcal{P}_a$ derives an attribute pseudonym t_a from attribute a via the shared secret K_a : $t_a = \{a\}_{K_a}$ where $\{x\}_y$ denotes symmetric encryption of plaintext x under key y . The publisher then attaches t_a to the advertisement message m_{adv} , and spreads the message to its neighbors $N(p)$ in G .

HASH CHAINS Flooding depends on a method for every node to identify already processed messages, e.g., identify duplicates due to parallel paths and loops. The same challenge applies to forest fire and random walks as well. Sender **IDs**, transaction **IDs** and **TTL** counters as common mechanisms cannot be used as they would allow an attacker to easily break anonymity. This thesis therefore proposes hash chains as a method to overcome this challenge. First, a hash chain does not reveal any information given just one element. However, a hash chain allows to link two element to the same chain. Second, a hash chain also allows to determine the distance between two elements of the hash chain. Finally, hash chains are efficient to calculate.

Using the introduced notation, the challenge can be illustrated as follows: flooding an advertisement m_{adv} that only contains t_a without global node **IDs** (anonymity) may cause duplicate messages and even routing loops. Figure 18 depicts these challenges in an unstructured **P2P** network without global node **IDs**. Therefore a transaction pseudonym h is required.

Example 15. Figure 18 (left) shows three forwarders. Forwarder f_1 sends the advertisement to f_2 , the advertisement then gets forwarded to f_3 . Due to limited neighborhood knowledge and the absence of global node **IDs**, f_3 forwards to its neighbor f_1 again. Thus f_1 depends on the transaction pseudonym / distance metric h and successor element h'' to detect loops. Figure 18 (right): shows an example where f_4 received the same advertisement from f_2 and f_3 . To detect duplicate advertisements, f_4 requires a transaction pseudonyms in this case as well.

This thesis combines a transaction pseudonym with distance metric within hash chains. An element of a hash chain is attached to every advertisement message. Relaying this message will cause this



Figure 18: Transactions pseudonyms for advertisements. Left: loop detection. Right: duplicate detection.

element to be incremented. Two advertisements then can be linked to same origin by verifying if one hash chain element is an increment of the other one.

A hash chain is initialized with a nonce—a transaction pseudonym—and provides a distance metric in combination with the chain length. Successive elements of the hash chain are derived by hashing the previous element. Furthermore, this mechanism does not leak the distance from the origin to any attacker, but rather the relative distance between two elements.

The hash chain approach yields another significant advantage over TTL counters: malicious insiders cannot claim an “earlier” chain element due to the one-way property of hash functions. That means, an attacker cannot offer a path with shorter distance to an attribute than the shortest one the attacker received itself. This impedes an attacker in mounting greyhole and blackhole attacks, i.e., steal traffic.

Given a cryptographic hash function $H(h) = h'$ that takes h as input and outputs h' , a hash chain is defined by all values h, h', h'', \dots derived by repeatedly applying H to its output. The function *inChain* tests if two hash values h_i and h_j belong to the same hash chain and have a maximum distance d_{\max} . In case both values belong to the same hash chain, the function returns the distance, otherwise 0.

$$\text{inChain}(h_i, h_j) = \begin{cases} k & \exists k \in [0, d_{\max}] : H^{(k)}(h_i) = h_j \\ -k & \exists k \in [0, d_{\max}] : H^{(k)}(h_j) = h_i \\ 0 & \text{otherwise} \end{cases}$$

The parameter d_{\max} is equivalent to a time-to-live and has to be set according to the estimated diameter of G , i.e., d_{\max} should be set higher than the estimated diameter. The following issues arise in case d_{\max} is set inappropriate:

d_{\max} TOO LOW If d_{\max} is set too low, duplicate advertisements are not detected as duplicates. As a consequence, multiple routing entries are stored by the node which failed to detect the duplicate, leading to potentially duplicate subscriptions and thus signaling overhead.

d_{MAX} TOO HIGH If d_{\max} is set too high, unnecessary processing overhead arises. Experiments on an Intel low power Core i5 CPU (Sandy Bridge) with 1.6 GHz, 4 GB of RAM, and Python 2.7.10 show that $2 * 500 = 1,000$ iterations of the the secure hash algorithm (SHA)-256 can be computed in 4.93 milliseconds ($\sigma^2 = 1.1$ millisecond). Even a very high limit, such as $d_{\max} = 500$, therefore causes only mild computational overhead. An extremely high d_{\max} in combination with short hash values may also lead to hash collisions. In this case, two hash chains are falsely considered as one, leading to an overlay separation for the related attributes.

The following estimates can be used for d_{\max} : Tor [44] uses three guards plus sender and receiver for a circuit. Three guards are the minimum to protect against a malicious insider threat. The topology including guards, sender, and receiver contains 5 nodes and therefore 4 hops. The parameter d_{\max} should be therefore greater than 4 to support basic overlays sufficiently large to protect anonymity. The Internet protocol (IP) in version 4 specifies 8 bit for the TTL header field—a maximum of 255 hops. To maintain compatibility with algorithms and strategies for the IP, d_{\max} should not be set higher than 255.

The publisher initializes a chain with a random h from the output domain of H , e.g., $H(N)$ given a nonce N , and attaches it together with pseudonym t_a to an advertisement message $m_{adv} = (t_a, h)$. Forwarders keep triples (t_a, h, v) in routing tables, where v is the last forwarder. The function *inChain* distinguishes multiple publishers, duplicates, and loops. Furthermore, they choose the shortest path in case of duplicates. When continuing to flood, forwarders increment the hash chain, i.e., derive h' from h and so on.

The Procedure *processAdvertisement* describes how forwarders process advertisements: the procedure takes the triple (t_a, h, v) from the advertisement, where v denotes the last sender of the advertisement. The tuple (w, T^{adv}) denotes the state of the forwarder, the own node *ID* w and the advertisement table T^{adv} . The forwarder then searches for an existing attribute record R_{t_a} given t_a in line 1. Now two cases are possible, the attribute pseudonym t_a is new to the forwarder (lines 2–10) or t_a is already known (lines 11–23).

If a forwarder w has not received pseudonym t_a before, w stores it in the routing table T^{adv} as (h, v) within a new record R_{t_a} (lines 3–4) and forwards (t_a, h') with $h' = H(h)$ to all neighbors except v (lines 5–8).

Otherwise, w already has a record R_{t_a} with one or more tuples (h^*, v^*) (line 11). For every tuple, it checks if the output of *inChain* (h, h^*) returns a non-zero result. This indicates that both advertisements belong to the same hash chain. Furthermore, in case h is a predecessor of h^* in the hash chain (line 12), a shorter path has been discovered and w replaces (h^*, v^*) by (h, v) (lines 13–15). If h is a successor of h^* (line 17), the forwarder discards the advertisement as duplicate (line 18). If *inChain* (h, h^*) returns 0 for all h^* (line 21), h belongs to a yet

Procedure processAdvertisement($(t_a, h, v), (w, T^{adv})$)

```

1  $R_{t_a} \leftarrow \text{findRecord}(t_a, T^{adv});$ 
2 if  $R_{t_a} = \emptyset$  then
3    $R_{t_a} \leftarrow \{(h, v)\};$ 
4    $\text{updateRecord}(t_a, R_{t_a}, T^{adv});$ 
5   for  $(u \in N(w) \setminus v)$  do
6      $m_{adv} \leftarrow (t_a, H(h));$ 
7      $\text{message}(u, m_{adv});$ 
8   end
9   return;
10 end
11 for  $(h^*, v^*) \in R_{t_a}$  do
12   if  $\text{inChain}(h^*, h) < 0$  then
13      $R_{t_a} \leftarrow R_{t_a} \setminus (h^*, v^*) \cup (h, v);$ 
14      $\text{updateRecord}(t_a, R_{t_a}, T^{adv});$ 
15     return;
16   end
17   if  $\text{inChain}(h^*, h) > 0$  then
18     return;
19   end
20 end
21  $R_{t_a} \leftarrow R_{t_a} \cup (h, v);$ 
22  $\text{updateRecord}(t_a, R_{t_a}, T^{adv});$ 
23 return;

```

unknown hash chains. Another publisher for the same t_a has been found, and w stores (h, v) as another tuple in the record R_{t_a} (lines 21+22).

AUTHENTIC ADVERTISEMENTS Pseudonyms must be signed, so that malicious insiders cannot create arbitrary advertisements and overload the system. So far, an advertisement consists of an attribute pseudonym and a hash chain element, e.g., (t_a, h) . As the hash chain element h is incremented by every passing node, only the attribute pseudonym t_a is signed. For that, the publisher executes a function $\text{sign}(t_a, sk_a)$ to obtain the signature $t_{a\text{sig}}^{sk_a}$.

$$m_{adv} = (t_a, t_{a\text{sig}}^{sk_a}, \text{cert}_a, h) \quad (44)$$

The advertisement is then extended to the message as shown in Equation (44): the attribute pseudonym, a signature of the attribute pseudonym, a certificate containing the public signature key together with key signature of the TTP, and the hash chain element. The certificate ensures that every node can verify the signature. Every verifying node is only required to know the public signature key of the TTP. The verification can be therefore performed offline, i.e., without contacting the TTP, and the system scales with the number of attributes, i.e., no key storage per attribute is required.

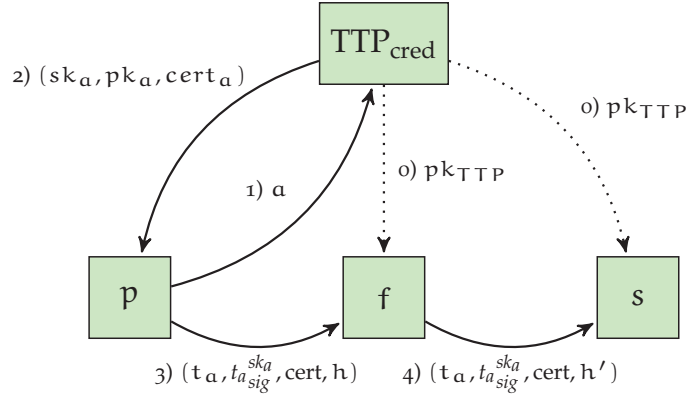


Figure 19: Signatures for advertisements: Distribution of **TTP** public key (o), request for and response for signature key (1,2), forwarding and message validation (3,4).

The key lifecycle is summarized in Figure 19. The key management building block ensured that all participants possess the public key of the **TTP** (step o). Upon joining the system, the publisher obtained signature key and certificate (steps 1, 2). Now every node can verify that an attribute is valid upon reception of an advertisement (steps 3, 4).

Every participant verifies advertisements by checking $cert_a$ and $t_{a_{sig}}^{sk_a}$. Hence, participants can detect non-authentic advertisements as well as duplicates. This check leaves a malicious insider with following options:

REPLAY An attacker can replay a message as it is. Every node receiving this message either ignores the advertisement if t_a is already known or adds a routing entry for the attacker, which is in compliance with the protocol.

ALTER h An attacker can alter the hash chain when replaying the advertisement. Picking a successive element of an existing hash chain will cause receiving nodes to ignore the advertisement in favour of advertisement with a preceding hash chain element. Picking a nonce will cause the receiving nodes to consider that the attacker has a path to another publisher. This will cause subscription for a to be forwarded to the attacker in addition to genuine publishers.

Example 16. Figure 20 explains the advertisement process with hash chains and key management. Publishers obtain $cert_a$ from the TTP_{red} , and create the advertisement as $m_{adv} = (t_a, t_{a_{sig}}^{sk_a}, cert_a, h)$ containing the encrypted attribute, hash value, signature, and certificate. They then flood the advertisement to all their neighbors. Each time a node receives the advertisement, it verifies if the signature is valid, i.e., t_a is an existing attribute and originated from an authorized publisher. When the node forwards this advertisement, it increments the hash chain ($h \rightarrow h' \rightarrow h'' \dots d_{max}$). Hence nodes with existing advertisement table entries for t_a can also verify if the new advertisement is fresh or belongs to the existing transaction.

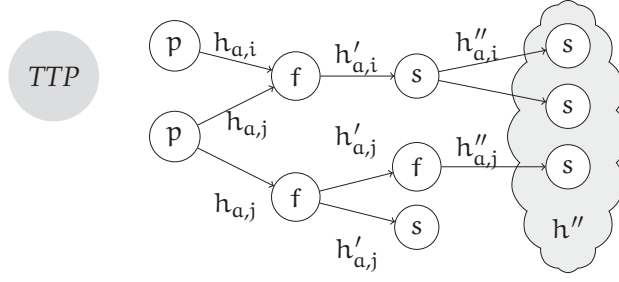


Figure 20: Distribution of advertisements. Forwarder increment hash chain elements ($h \rightarrow h' \rightarrow h''$) and merge advertisements when forwarding.

SUMMARY This section explained how flooding can be used as a basis for anonymous attribute distribution. For that, transaction pseudonyms and a distance metric were combined in the form of a hash chain for the first time.

ANONYMITY The anonymity of publishers is preserved here as no node IDs are transmitted. Moreover, not even the distance to the publisher is transmitted due to the use of hash chain. Given two advertisements, the hash chain merely reveals which advertisement took the shorter path, as well as how much shorter the path is.

CONFIDENTIALITY Attributes are kept confidential as well as attribute-specific key material is required to link an advertisement to an attribute.

MINIMAL OVERHEAD Flooding as the foundational mechanisms ensures functional correctness such that attribute knowledge is distributed to all nodes. The hash chains in the sense of a transaction pseudonym allow nodes to detect and drop duplicates. This ensures that the signaling overhead is kept low. The hash chains in the sense of a distance metric allow nodes to always pick the shortest path to a publisher. This ensures that the signaling overhead for the content distribution building block is kept low as well.

SCALABILITY The routing tables maintained by nodes grow linear with the number of attributes. Digital signatures are used to ensure the authenticity of advertisements, i.e., allow all nodes V to verify m_{adv}^a originated from a genuine publisher $p \in \mathcal{P}_a$. As a consequence, malicious insiders cannot create false attributes to overload the system.

In summary, the presented approach fulfills all requirements except for minimal overhead. While the approach ensures that the signaling overhead is kept low, the underlying concept may cause the transmission of duplicates. That is, nodes may receive an advertisement concerning the same attribute and publisher from two or more neighbors. The following two sections therefore propose alternatives based

on the concepts of forest fire and random walks that do not rely on flooding.

4.3.2 Forest fire

Forest fire is a probabilistic graph traversal algorithm [88, 89]. This thesis proposes a novel method that incorporates forest fire to distribute attribute knowledge within an overlay network. This method can be generalized to *anonymous bootstrapping* for further applications. This method is used in two variations:

1. The attribute knowledge is distributed to all nodes V .
2. The attribute knowledge is distributed to a subset $V^* \subseteq V$, i.e., it is distributed sparsely in G .

This section first introduces the preliminaries of forest fire. Second, it elaborates on the first variation, how forest fire can be applied to distribute attribute knowledge. Third, it shows the second variation, how attribute knowledge can be distributed sparsely with privacy preserving TTL counter.

As forest fire is a probabilistic algorithm that may not traverse all edges, unlike flooding, which resembles a breadth-first search (BFS) from the graph traversal perspective. Transferred to the distribution of attribute knowledge, forest fire may therefore distribute this knowledge sparse compared to flooding. This thesis adds a TTL counter to forest fire to further enforce the sparseness of the node coverage.

FOREST FIRE GRAPH TRAVERSAL The basic forest fire algorithm according to Leskovec et al. [89] works as follows: the system requires two parameters, the forward burning probability p and the backward burning ratio r . Assuming a graph $G = (V, A)$ with nodes V and arcs A , a start node called ambassador node $w \in V$ is selected. The ambassador node picks x neighbors $V' \subseteq N(w)$. For that, x is drawn from a binomial distribution with mean $(1 - p)^{-1}$. The higher p , the more neighbors will be selected. The node then draws x random neighbors from $N(w)$. The backward burning ratio r defines that $x \div r$ incoming neighbors, i.e., neighbors from $N^+(w)$, are selected. Correspondingly, $1 - (x \div r)$ outgoing neighbors from $N^-(w)$ are selected. The nodes V' are then visited, and the same algorithm is applied recursively with every node $v \in N'$ as with w . The traversed arcs (w, v) are marked as burned—removed from A —and cannot be used again, effectively reducing $N(w)$. Nodes terminate the graph traversal when no more arcs can be burned.

To use forest fire for attribute distribution, the following modifications to the previously described attribute distribution have to be made:

1. Upon receiving an advertisement by a node v , the node does not forward the advertisement to all neighbors $N(v)$. The node rather draws x random neighbors from $N(v)$.

2. Every publisher p initializes the attribute distribution by selecting itself as “ambassador node w ” and follows the instruction as above.
3. Compared to the forest fire graph traversal, the backward burning ratio r is not used as the basic overlay G is undirected.
4. All other mechanisms, such as the hash chain, are used as before.

In comparison with the attribute distribution based on flooding, the attribute distribution based on forest fire should cause less signaling overhead. The next chapter analyzes the arising signaling overhead of both approaches in detail.

SPARSE VARIATION While the introduced approach with forest fire distributes attribute knowledge for a among all nodes V in the basic overlay G , it still causes signaling overhead. In particular in case no ($S_a = \emptyset$) or only few subscribers ($S_a \ll V$) interested in a exist, the signaling overhead can be further reduced. For that, the attribute knowledge is only distributed to a subset $V * a \subseteq V$. This reduces the signaling overhead as well as the storage overhead—nodes $V \setminus V * a$ do not have to store any information about a . However, this may cause subscribers to have to “search” for this attribute knowledge. The following Section 4.3.3 will answer the question “How can subscribers search for attribute knowledge?” while the publisher perspective is here covered first.

In order to distribute the attribute knowledge only to $V * a$ a termination criteria for the attribute distribution is required. A time to live (**TTL**) for advertisements can be used for that. It is important to note the previously introduced hash chains cannot replace a **TTL** as with hash chains two advertisements are required to calculate the distance.

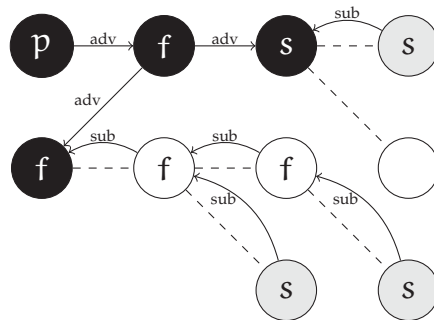


Figure 21: Attribute localization with forest fire.

Example 17. Figure 21 shows an example of the attribute distribution based on forest fire and **TTLs**. The publisher p initiates the attribute distribution with $\text{TTL} = 2$ and burns the black nodes. This causes signaling costs of 3 messages. However, 3 subscribers s marked in gray do not receive the advertisement and therefore have to search for it. This causes additional signaling

costs of 5 messages. Attribute knowledge based on flooding would have caused signaling costs of 9 messages, an overhead of 1 message compared to variation with forest fire and TTL.

TTLs are a suitable for a probabilistic attribute distribution, but TTLs would not sufficiently protect the anonymity in case of a deterministic attribute distribution as in the case of the method based on flooding. If a TTL was used with flooding, malicious insiders could trivially use the TTL as distance to the publisher and triangulate the position of the publisher.

PRIVACY-PRESERVING TTL While a TTL in combination with a probabilistic attribute distribution does not immediately allow a malicious insider to break anonymity, further measures should be taken to prevent such attackers from inferring small publisher candidate sets. If the TTL was initialized by a publisher p with a system parameter, e.g., d_{\max} , and strictly decremented by 1 with every hop, the malicious insider could trivially infer the distance to p as $(d_{\max} - \text{TTL})$. That would, in particular for short distances, leave only few nodes as potential publishers.

Random noise in combination with local information overcomes this issue. The initialization is performed as follows:

Every publisher $p \in \mathcal{P}_a$ draws a random number X_p^a from the Weibull distribution [166]. A parameter setting of $\lambda \geq 1$ and $k > 1$ ensures a preference of small values according to the probability density function of the Weibull distribution as given in Equation (45). The selection of the Weibull distribution in combination with this parameter range causes the density mean of the random variables to be lower than the median—preference of small values—while all random variables are positive (cf. Equation (45)) and a long tail in the distribution is avoided. For the TTL, drawing small random numbers, i.e., $X_p^a \ll d_{\max} \ll 255$, is sufficient as no benefit arises from drawing numbers exceeding the graph diameter— $X_p^a = d_{\max}$ would already guarantee a TTL high enough to render every node in V a potential publisher under all condition. To enforce the preference of the random variable X_p^a to be within interval $X \in [0, d_{\max}]$, $d_{\max} \div 2$ can be used as desired median ($E(X) = \lambda\Gamma(1 + 1 \div k)$) for the distribution.

$$f(x; \lambda, k) = \frac{k}{\lambda} \times \left(\frac{x}{\lambda}\right)^{k-1} \times e^{-(x/\lambda)^k}, \quad x \geq 0 \quad (45)$$

$$\text{TTL}_p^a = \lceil X_p^a \rceil + |N(p)| \quad (46)$$

As an example, Figure 22 depicts the density function of Weibull distribution for the interval $[0, 5]$ with the parameters $\lambda = 2.0$ and $k = 2.0$. Increasing the scale λ will increase the expectation values, whereas increasing k will reduce the variance.

The malicious insider may still guess the assignment of X_p^a , e.g., with a statistical disclosure attack [33]. The addition of local information prevents the malicious insider from such guessing. By intuition, having a large neighborhood, i.e., a large set $N(p)$, increases the chances of having received the advertisement from one of these

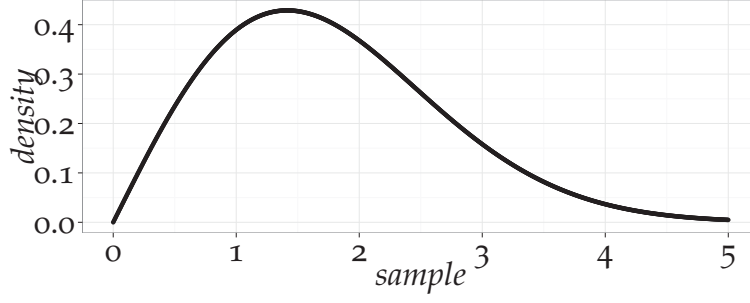


Figure 22: Density plot of the Weibull distribution, $\lambda = 2, k = 2$

neighbors. Incrementing X_p^a by the neighborhood size therefore reflects these chances. In summary, the publisher initializes the **TTL** as given in Equation (46).

$$m_{adv} = (t_a, t_{a_{sig}}^{sk_a}, cert_a, h, TTL) \quad (47)$$

Together with attribute pseudonyms, signature, hash chain, and **TTL** the advertisement message m_{adv} becomes a 5-tuple as given in Equation (47).

SUMMARY This section proposed a method based upon forest fire to distribute attribute knowledge with potentially less signaling overhead than the flooding-based mechanism. The behaviour of the forest fire method can be controlled with the forward burning probability p . In addition, a **TTL** can be set to distribute the attribute sparsely. With sparse distribution, the signaling overhead is reduced, but subscribers may have to search for nodes with the desired attribute knowledge. To prevent anonymity attackers from reasoning about the distance to the publisher given the **TTL**, a privacy-preserving **TTL** variation was introduced that uses noise and local properties of nodes to hide their distance. The following section proposed a third method to distribute attribute knowledge and introduced the search method for subscribers.

4.3.3 Double random walk

Like forest fire, the random walk is a probabilistic graph traversal algorithm. While flooding resembles a breadth-first search (**BFS**), the random walk resembles a depth-first search (**DFS**). The forest fire can be classified in-between both categories. Like forest fire, a random walk can be used to distribute attribute knowledge within the basic overlay. In order for subscribers to discover this attribute knowledge, they might be required to perform another random walk by themselves. This thesis proposes this approach as a *double random walk*.

RANDOM WALK GRAPH TRAVERSAL A random walk [3] works as follows: assuming a graph $G = (V, E)$ with nodes V and edges E , a

start node $w \in V$ is selected. This node picks a random neighbor $v \in N(w)$ uniformly at random. The node v is then visited and continues recursively by picking another neighbor $v' \in N(v) \setminus w$. The traversed edges (w, v) are marked as visited—removed from E —and cannot be used again. In case of $N(v) \setminus w = \emptyset$, the random walk terminates. A random walk may employ backtracking to overcome the algorithm termination by empty neighborhood sets. In case of $N(v) \setminus w = \emptyset$, the random walk would backtrack to a previous node w with a non-empty set of not selected neighbors, and select another neighbor $v' \in N(w) \setminus v$.

This mechanism can be applied to the distribution of attribute knowledge—similar to the forest fire-based method—as well as for the discovery of attribute knowledge by subscribers.

DISTRIBUTION OF ATTRIBUTE KNOWLEDGE VIA RANDOM WALKS

A random walk with backtracking eventually traverses the complete basic overlay. This may cause higher signaling costs than flooding due to the backtracking. However, with a privacy-preserving **TTL** as introduced in the previous section, the random walk can be used as a basis to spread attribute knowledge sparsely as well. For that, the publisher sets the **TTL** of the advertisement as given in Equation (46) as well. Compared to the attribute distribution via forest fire, the constant assignment of $x = 1$ replaces the stochastic assignment of x . Furthermore every node v performs backtracking if there are no remaining unburnt neighbors. For that, v send the advertisement back to the originator with the hash chain and **TTL** the message was received.

A node v detects that an advertisement backtracked if all of the following conditions hold:

1. v receives an advertisement (t_a, h, TTL) with **TTL** > 0 from node w .
2. The hash value has been sent out before, i.e., an advertisement record with (h^*, u) exists, and $H(h^*) = h$.

In this case, the advertisement is not a duplicate and cannot be discarded.

DISCOVERY OF ATTRIBUTE KNOWLEDGE VIA RANDOM WALKS

If the **TTLs** of all circulating advertisements for an attribute reach 0 during the distribution of attribute knowledge via forest fire or a random walk, not all subscribers might be reached as shown by Figure 23. In this example, p initiated a random walk with an advertisement and $TTL = 3$. In this case, the left unreachable subscribers attempt to discover attribute knowledge. For that they initiate a random walk with a subscription message. This subscription must not contain a **TTL** limit, i.e., $TTL_s^a = \infty$. That means the random walk will only terminate under the conditions of either a successful discovery of attribute knowledge or a backtrack to the subscriber without unburnt neighbors.

CONNECTING THE OVERLAY In case the attribute overlay \mathcal{M}_a contains multiple publishers, every publisher is required to initiate a random walk like a subscriber; the publishers therefore subscribe to each other. This ensures \mathcal{M}_a is connected, in particular that Constraint (48) is fulfilled. Note that Constraint (48) implies Constraint (49) by connecting two paths $path(p_1, s), path(s, p_2)$ each.

$$\forall s \in \mathcal{S}_a, \forall p \in \mathcal{P}_a : \exists path(p, s) \quad (48)$$

$$\forall p_1, p_2 \in \mathcal{P}_a : \exists path(p_1, p_2) \quad (49)$$

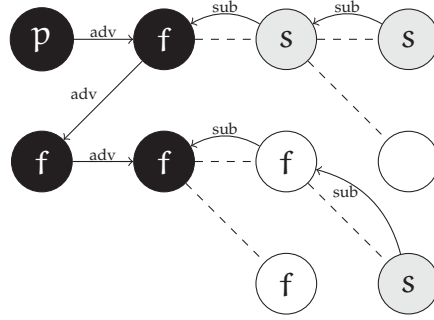


Figure 23: Attribute localization with double random walks. $TTL_p^a = 3$.

Example 18. Figure 23 shows an example where publisher p distributes the attribute knowledge via a random walk of length 3, i.e., causing signaling costs of 3 messages. To discover this attribute knowledge, the subscribers s marked in gray have to perform random walks with signaling costs of 4 messages. The total signaling costs are therefore 7 messages.

SUMMARY This section presented three mechanisms for distributing and discovering attribute knowledge within unstructured **P2P** overlays based upon three graph traversal methods. Furthermore, the concept of hash chains was applied to the distribution of attribute knowledge to preserve participant anonymity. A privacy-preserving **TTL** can be used in addition to spread the attribute knowledge sparsely in the basic overlay.

Flooding as a basis to distribute attribute knowledge works deterministically and reaches all nodes of the basic overlay. This ensures that the resulting attribute overlays are hop-minimal. This causes the signaling overhead for the content distribution to be low. However, during the attribute distribution, every edge is traversed at least once and up to twice, potentially causing signaling overhead. Furthermore, every node must store information about every attribute.

Forest fire as a basis for attribute distribution burns edges to neighboring nodes like a fire, causing each edge to be used at most once. That may cause potentially less signaling overhead. However, the attribute overlays are not guaranteed to be hop-minimal anymore. This approach is also stochastic in nature as not all adjacent edges of a node are burned, but only a random subset. The size of this subset

can be controlled with a system parameter. As a second variation, this mechanism can be equipped with a **TTL**, resulting in a termination of the mechanism before all nodes are reached—given the **TTL** is sufficiently low. Consequently, the attribute knowledge is distributed sparsely in the basic overlay. This potentially reduces the signaling overhead due to the not-traversed edges. Furthermore, the storage space per node is reduced in case the node does not receive the attribute knowledge.

For subscribers to obtain the attribute knowledge in the latter case, an active discovery is required. For that, a random walk with a subscription message was proposed. Such a random walk can be also used for the distribution of attribute knowledge instead of the forest fire. In this case, the random walk is performed with a **TTL** attached to the advertisement such that the relaying of the message terminates before all nodes are traversed. In summary, the following combination to distribute attribute knowledge can be performed based upon the graph traversal method:

1. Flooding + subscriptions
2. Forest fire + subscription
3. Forest fire with **TTL** + random walk
4. Random walk with **TTL** + random walk (double random walk)

These variations are expected to differ in signaling overhead given the topology of the basic overlay and the number of subscribers per attribute. The first two variations cause the signaling overhead to arise due to advertisement messages. The latter two variations shift over more signaling overhead to subscription messages.

For all variations, hash chains were proposed as a privacy-preserving combination of transaction pseudonym and distance metric. Advertisement messages require a transaction pseudonym to distinguish multiple publishers from duplicate message. A distance metric helps to select the shortest path to a publisher. A hash chain incorporates both, but does not reveal the absolute distance to a publisher; the hash chain only reveals the distance difference between two messages.

In case an early termination for the distribution of attribute knowledge is desired, a **TTL** for messages is required. To prevent such a **TTL** to reveal the distance to the publisher, and thus possibly the position and node **ID** of the publisher, noise is required. This section proposed to use local properties as well as noise for that. The local knowledge is constituted by the size of neighborhood—more neighbors increase the likelihood of a message having originated via a longer path via one of the neighbors. A random sample in addition to the neighborhood size prevents anonymity attackers from linking a message based upon the **TTL** and neighborhood size to the publisher.

4.4 MATCHING

AnonPubSub uses binary comparison of pseudonyms for matching as introduced in [37]. For that, subscribers prepare the attribute pseudonyms for their topics once they obtained key material, and compare the pseudonyms of received advertisements with the prepared pseudonyms. In case of a successful match, subscribers send a subscription message along the reverse path of the advertisement. All nodes merge subscriptions whenever possible. For that, they maintain a subscription table T^{sub} to keep track of subscribed attribute pseudonyms. In case a subscription is received—or the node intends to subscribe itself—it adds the subscribing node ID the attribute pseudonym record in the table. If the record does not exist, subscription messages are sent out towards all publishers. Otherwise, no further actions are necessary, i.e., subscriptions have been merged.

As exemplified in Figure 24, subscribers S_a obtain attribute pseudonym t_a via K_a for a as publishers do. When a subscriber s receives advertisement triples (t_x, h', v) (attribute localization), it compares t_x to t_a (step 1) and subscribes with the shortened tuple (t_a, s) backwards the advertisement path (steps 2, 3).

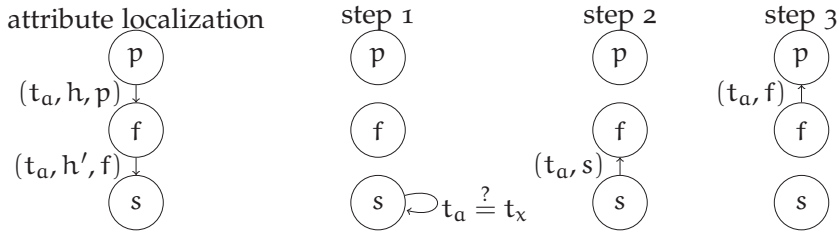


Figure 24: Messages from attribute localization to the left as reference. Step 1: a subscriber compares interest t_x with advertised attribute t_a . Step 2: the subscriber subscribes towards f . Step 3: f forwards the subscription to p .

Once the t_x from the triple is confirmed to match t_a , the subscriber s joins the mesh \mathcal{M}_a via a subscription (Figure 24 steps 2, 3).

SUBSCRIPTIONS Subscriptions are sent back the reverse path of the advertisements. A subscriber adds tuple (t_a, s) to the subscription table, where s is the subscribing node itself. Moreover, it sends a subscription message $m_{sub} = (t_a)$ towards the origin of the advertisement.

Procedure `processSubscription` lists the subscription process: whenever a forwarder w receives a subscription (t_a) from neighbor v , w updates its subscription table T^{sub} appending v to the record $R_{t_a}^{sub}$ (lines 10+11). If there is no subscription record yet (line 2), w obtains

the corresponding advertisement record $R_{t_a}^{adv}$ (line 3) and forwards the subscription to advertising node v^* , i.e., towards every publisher. This approach integrates well with attribute localization via flooding, forest fire, and random walks.

Procedure processSubscription($(t_a, v), (w, T^{adv}, T^{sub})$)

```

1  $R_{t_a}^{sub} \leftarrow \text{findRecord}(t_a, T^{sub});$ 
2 if  $R_{t_a}^{sub} = \emptyset$  then
3    $R_{t_a}^{adv} \leftarrow \text{findRecord}(t_a, T^{adv});$ 
4   for  $(h^*, v^*) \in R_{t_a}^{adv}$  do
5      $m_{sub} \leftarrow (t_a);$ 
6      $\text{message}(v^*, m_{sub});$ 
7   end
8   return;
9 end
10  $R_{t_a}^{sub} \leftarrow R_{t_a}^{sub} \cup v;$ 
11  $\text{updateRecord}(t_a, R_{t_a}^{sub}, T^{sub});$ 
12 return;

```

4.5 COMMUNITY MANAGEMENT

Community management maintains an established attribute overlay, e.g., compensates node churn and distributes the processing and signaling overhead fairly among nodes. Like other building blocks, community management also has to ensure confidential communication as well as preventing a malicious insider $\mathcal{I}\mathcal{A}$ from abusing community management, e.g., from stealing messages.

This section first introduces how an attribute overlay is maintained under churn—nodes entering/leaving the overlay as well as nodes failing. Next, it is explained how community management ensures that an attribute overlay always remains connected and how messages exchanged within the attribute community can be protected to prevent information leakage, e.g., to the global observer threat $\mathcal{G}\mathcal{A}$. This section then introduces a novel mechanism for anonymous inter-overlay optimization that distributes the forwarder role fairly among nodes.

4.5.1 Node churn and overlay connectivity

Whenever a publisher leaves the system, it un-advertises its attributes. Similarly, a subscriber sends an un-subscription message the same path as the subscription to leave the system. The same applies to overlay forwarders, which subscribed or advertised on behalf of other nodes. For that, advertise and subscribe messages are extended with un-advertise and un-subscribe. Both messages are forwarded the same path as the preceding subscribe and advertise messages. This is possible as forwarders can lookup the next hop in their advertisement / subscription tables before deleting the entry.

Process `processUnadvertisement` lists the un-advertisement procedure. First the node w obtains the advertisement record for $R_{t_a}^{adv}$ (line 1). Then it removes the tuple (h, v) of the un-advertising node v (lines 2+3). Afterwards, if the advertisement record is empty (line 4), v forwards the un-advertisement to all neighbors (lines 5–8).

Procedure `processUnadvertisement` $((t_a, h, v), (w, T^{adv}))$

```

1  $R_{t_a}^{adv} \leftarrow \text{findRecord}(t_a, T^{adv});$ 
2  $R_{t_a}^{adv} \leftarrow R_{t_a}^{adv} \setminus \{(h, v)\};$ 
3  $\text{updateRecord}(t_a, R_{t_a}^{adv}, T^{adv});$ 
4 if  $R_{t_a}^{adv} = \emptyset$  then
5   for  $(u \in N(w) \setminus v)$  do
6      $m_{unadv} \leftarrow (t_a, H(h));$ 
7      $\text{message}(u, m_{unadv});$ 
8   end
9 end
10 return;

```

Process `processUnsubscription` lists the un-subscription procedure. The node w obtains the subscription record for $R_{t_a}^{sub}$ (line 1). Then it removes the un-subscribing node v from the record (lines 2+3). Afterwards, if the subscription record is empty (line 4), v forwards the un-subscription to advertising nodes v^* (lines 6-9) found in the corresponding advertisement record $R_{t_a}^{adv}$ (line 5).

Procedure `processUnsubscription` $((t_a, v), (w, T^{adv}, T^{sub}))$

```

1  $R_{t_a}^{sub} \leftarrow \text{findRecord}(t_a, T^{sub});$ 
2  $R_{t_a}^{sub} \leftarrow R_{t_a}^{sub} \setminus v;$ 
3  $\text{updateRecord}(t_a, R_{t_a}^{sub}, T^{sub});$ 
4 if  $R_{t_a}^{sub} = \emptyset$  then
5    $R_{t_a}^{adv} \leftarrow \text{findRecord}(t_a, T^{adv});$ 
6   for  $(h^*, v^*) \in R_{t_a}^{adv}$  do
7      $m_{unsub} \leftarrow (t_a);$ 
8      $\text{message}(v^*, m_{unsub});$ 
9   end
10 end
11 return;

```

This processing of un-advertisement and un-subscriptions may lead to situations, where a forwarder leaves the basic overlay, but publishers and subscribers are still present. These remaining nodes may no be disconnected from the attribute overlay. In this situations these nodes either have to wait for advertisements to be re-distributed, e.g., assuming an epoch-based advertisement cycle, or rely on gossiping. With gossiping [61], remaining nodes inquire their neighbors after the un-advertisement for t_a has been processed, if the neighbor is still in possession of an advertisement for t_a . Then, the inquiring node can act as it received the advertisement from the neighbor. The

attribute overlay can then be reconnected using the procedures from the previous sections.

ATTACK BASED ON CHURN Churn also poses a threat to subscriber anonymity due to the global observer threat $G\mathcal{A}$. It is vital to consider this threat as it has been used manyfold times to break anonymization services. Section 2.3.6.2, page 23, lists such attacks.

Churn is observable in particular when a node joins or leaves the basic overlay G and the attribute overlay \mathcal{M}_a simultaneously. The attacker can use these observations to tell forwarders from subscribers, and therefore de-anonymize subscribers. Similar attacks on other anonymization services have been reported as *intersection attack* [119].

The following illustrates such attacks with an example: Figure 25a depicts the case of the shaded node s_2 joining at time $t + 1$. Initially, only s_1 is connected to p . Thus, $\mathcal{S}_a^t = \{s_1\}$, $\mathcal{F}_a^t = \{f_1, f_2\}$, and $path_a^t(p, s_1) = (p, f_1, f_2, s_1)$ as s_2 has not joined at time t yet. Once s_2 joins, it directly connects with p as $path_a^{t+1}(p, s_2) = (p, s_2)$. However, $path_a^{t+1}(p, s_1)$ should now become (p, s_2, s_1) rather than (p, f_1, f_2, s_1) . As a result, both subscribers s_1, s_2 are exposed as leaf nodes at $t + 1$, whereas ideally only s_1 should be exposed.

While advertisement and subscription process causes this situation to be repaired, i.e., s_1 reconnects via s_2 , a notification may still flow from p to s_1 right after s_2 connected, but before s_1 reconnected via s_2 . If the global observer threat witnesses this, it becomes evident that s_2 is a subscriber.

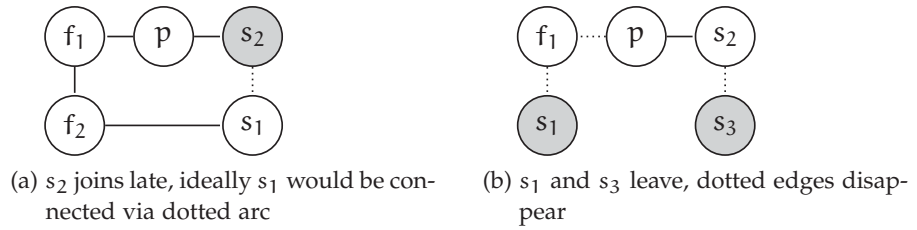


Figure 25: Effects of churn.

Figure 25b depicts the case of the shaded nodes s_1, s_3 leaving at $t + 1$. As s_1 leaves \mathcal{M}_a , f_1 may also leave \mathcal{M}_a as it is no longer required to relay message at $t + 1$. This “ripple effect” allows the global observer to reason that $f_1 \in \mathcal{F}_a$ and $f_1 \notin \mathcal{S}_a$. Likewise, when s_3 leaves \mathcal{M}_a at $t + 1$, s_2 will remain in \mathcal{M}_a as it is interested in the notifications for a . The $G\mathcal{A}$ reasons that $s_2 \in \mathcal{S}_a$ and $s_2 \notin \mathcal{F}_a$.

In summary, node churn exposes subscribers over time. Furthermore, churn affects the overlay topology, which in consequence has impacts on the overlay optimization that might lead to further exposure. Thus, the effects of churn must be carefully considered when introducing new anonymity-enhancing overlay modifications.

MULTIPLE PUBLISHER In case there are multiple publishers for the same attribute, additional measures are required to ensure mesh \mathcal{M}_a remains connected in case publishers are joining and leaving. Assum-

ing two publishers p_1 (transaction pseudonym h_1), and p_2 (pseudonym h_2), and further assuming p_2 joins after p_1 , then p_2 or a node v between both publishers recognizes the distinct transaction pseudonyms h_1 and h_2 . Hence, this node forwards the advertisement with h_2 towards p_1 and advertisement with h_1 towards p_2 . Therefore both publishers learn that at least one other publisher exists and thus also act as forwarders by forwarding subscriptions towards the other publisher.

HEARTBEATS AND PADDING The attacker global observer threat may infer the overlay topology through message content and message flow. Confidential links for the basic membership as well as message padding to render message types indistinguishable overcome this issue. As an estimation, 73 bytes + underlay headers suffice for message type, pseudonym, hash chain, and signature. Finally, reoccurring messages, e.g., heartbeats, piggy-back these 73 bytes padded management messages to prevent the leakage of message flow. In comparison, IPv4 and UDP headers add up to 28 bytes. An Ethernet frame can take 1,550 bytes of such payload, with a minimum payload of 46 bytes. The padded heartbeat via UDP over IPv4 therefore exceeds the minimum frame size by mere 55 bytes.

In summary, this section has addressed challenges and solutions regarding the anonymity and availability requirements of the community management building block. Nevertheless, attribute overlays may still degenerate over time in the sense that churn may cause overlay paths to increase in length. Moreover, long-living nodes may take the majority of the forwarding workload over time. The following section proposes a solution for this challenge by distributing the workload among nodes.

4.5.2 *Inter-overlay optimization with Bloom filters*

The construction of attribute overlays presented in the previous section includes nodes as pure forwarders, which neither take the publisher nor the subscriber role. Such forwarders may get overloaded with processing overhead as well as signaling overhead for two reasons: first, such nodes may become forwarders frequently due to high connectivity. Second, nodes being part of the overlay may maintain forwarder roles more often, whereas recently joined nodes may hardly become involved.

To reduce the overhead, the overloaded forwarder has to hand over the forwarder role for one or more attributes to another node. This thesis proposes such an inter-overlay optimization approach while protecting the anonymity of nodes. The proposed approach establishes random ad-hoc communities that exchange load information via Bloom filters. Overloaded nodes then attempt to identify other nodes acting as forwarder for one of their attributes via a Bloom filter, and hand over their forwarder role.

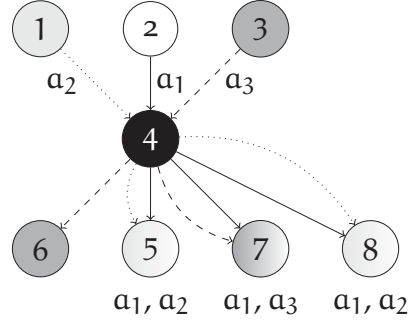


Figure 26: Example with 3 attributes: a_1 in white/solid, a_2 in light gray/-dotted, and a_3 in dark gray/dashed. Node 4 is overloaded.

Example 19. Figure 26 depicts an example with 8 nodes and 3 attributes. The figure shows a section of the basic overlay, given the local knowledge of node 4, with different line types representing the 3 attribute overlays. Attribute a_1 uses white nodes with solid arcs. Attribute a_2 uses light gray nodes with dotted arcs. Attribute a_3 uses dark gray nodes with dashed arcs. Node 4 is forwarder for all 3 attributes, but neither publisher nor subscriber for any of these attributes.

COMMUNITY FORMATION To perform inter-overlay optimization, first a community must be established. A random initiator establishes a community greedily. The community formation consists of two steps: first the establishment of a set of nodes participating in the community. Second, establishment of an overlay structure among this set of nodes.

To become an initiator, every node v waits for random period X after the last community it participated in has been dissolved. If the node never participated in a community, e.g., because the node just joined the basic overlay G , the node immediately waits for a random period X . Then the node inquires all basic overlay neighbors $N(v)$ to join the community. If as subset $N'(v)$ of sufficient size $|N'(v)| \geq 3$ accept, v is the leader of this community. Otherwise, v dissolves the partial community $N'(v)$. Every node is at most part of one community.

Figure 27 shows an example where node 4 acts as an initiator. The nodes 1 – 3, 5 – 8 are members of the same optimization community but members of different attribute overlays (Figure 26).

When the overlay optimization is complete, node 4 dissolves the community. All nodes independently start the process over again—they are invited into a community or attempt to initiate a community after a random time interval has passed.

After the set V' of nodes participating in the community has been established, an appropriate topology has to be established to allow communication within the group. While all nodes $v' \in V'$ are connected to the initiator, such a star topology would enable the initiator to learn each community member's load status (malicious insider

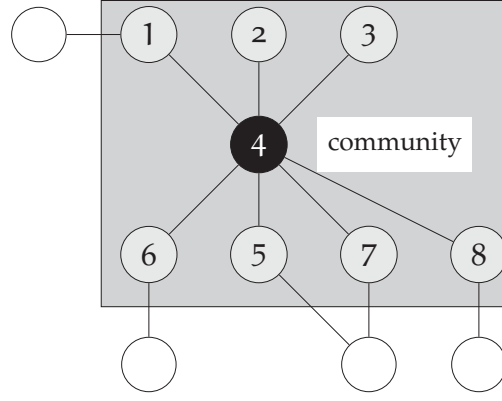


Figure 27: Inter-overlay optimization community: initiator is node 4, nodes 1, ..., 8 participate.

threat). The initiator v therefore connects all nodes in a directed ring. A set of arcs A defines this topology. An arc is a directed connection between two nodes. The initiator informs each community member about the outgoing arc of this member according to the rules stated in Equations (50), (51), and (52). It remains up to initiator to define the order of nodes, e.g., by the sequence in which nodes joined V' .

$$A_0 = \{\} \quad (50)$$

$$\forall v'_i \in V', 1 \leq i < |V'| : A_i = A_{i-1} \cup (v'_i, v'_{i+1}) \quad (51)$$

$$A = A_{|V'|-1} \cup \{(v, v'_1), v'_{|V'|}, v\} \quad (52)$$

Figure 28 depicts the community ring given the example from Figure 26.

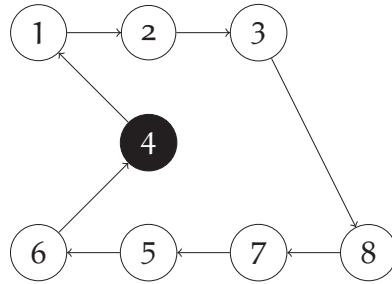


Figure 28: Inter-overlay optimization community connected as a ring with initiator 4.

COMMUNITY OPTIMIZATION The community optimization is the second phase of inter-overlay optimization. After the community has been established, a joint load status for the community is formed and distributed among all members. This status provides every node with an indicator if underloaded nodes are present, and if another node may already forward one of the own attributes. Nodes may then circulate attribute handover request, which may then be answered by handover responses. In case of a response, both, requester and responder, establish a direct community to handover the attribute.

Establishing the joint load status requires all nodes to reveal the attribute pseudonyms they forward. To prevent malicious insiders from learning this information, a privacy-preserving aggregation scheme is used: first, a counting Bloom filter is used as data structure. Second, the initiator initializes this filter with random noise, which is removed after all community members added their load status. The counting Bloom filter as a probabilistic data structure only stores hashes of the attribute pseudonyms such that the filter cannot be reversed to an attribute pseudonym unless the pseudonym is known. The filter also only reveals if an attribute pseudonym is likely to be present. A counting Bloom filter also gives an indication how often an attribute pseudonym might be present. Finally, the counting Bloom filter also allows to remove entries again. This allows nodes to use one filter as continuous data structure—nodes that take over an attribute add an entry to the filter; nodes that hand over an attribute remove the entry from the filter.

The circular communication within the community is performed according to the following steps:

1. The initiator $v_{i=0}$ creates a counting Bloom filter \vec{bf} , adds random noise r , and forwards the filter as a message to its community neighbor v_{i+1} . This message acts as a *token* on the ring. Only the initiator creates the token, all nodes wait, modify, and pass on the token.
2. Every community node adds its own load status to the counting Bloom filter. For that, the node adds all its attribute pseudonyms to the Bloom filter.
3. After every node added its load status, i.e., the counting Bloom filter circled once through the community, v_0 adds its own load status and removes the noise r . The noise ensured that every node, including v_0 never had access to the load status of few or even only one node. The anonymity set regarding linking the information of the counting Bloom to a node are therefore all community members.
4. The initiator v_0 complements the counting Bloom filter with an empty handover request and an empty handover response, and circles the new message through the community.
5. Every node receiving the message with empty handover request may replace the empty handover request with an attribute it wishes to handover. This node is called overloaded node v_{over} . The overloaded node then continues to circulate the message to the next neighbor.
6. Every node receiving the message with a handover request and empty handover response may accept the request and set the handover response. This node is called underloaded node v_{under} . The underloaded node then continues to circulate the message to the next neighbor.

7. Once v_{over} receives the message with its own handover request, it sets the handover request to empty again. If the handover is non-empty, it establishes direct communication with v_{under} . It then sets the response to empty. In any case, it continues to circulate the message.
8. The steps 5–7 are repeated until the initiator observes two empty handover requests in sequence.
9. The initiator dissolves the community.

If a node v wants to leave the community prematurely, the v removes its attribute pseudonyms from the counting Bloom filter and informs the initiator v_0 . The initiator then either dissolves the community immediately (community becomes too small), or holds the token during the next pass, reconnects the overlay without v , releases the token, and informs v . If a node v fails, v_0 dissolves the community after not receiving the token within a given timeout again.

With step 1, the initiator v_0 creates the message structure to be circled within the community. The message consists of a counting Bloom filter, a handover request block, and a handover response block.

Counting Bloom filters [53] (cf. Section 3.6.4, page 64) allow for the removal of entries as opposed to normal Bloom filters. This enables nodes to adjust their load status in the Bloom filter, e.g., remove an attribute they no longer forward for. As a result, a counting Bloom filter also allows to count how often a certain entry is present in the Bloom filter at most.

To test how often an attribute is present, the following test is performed: first, a node calculates a new Bloom filter solely for the attribute a to be tested (Equation (53)). Next, the node compares all 1-positions of the new Bloom filter with the counting Bloom filter, and identifies the position i with lowest number in the counting Bloom filters (cf. Equation (54)). Then $\vec{bf}_{load}[i]$ is the upper boundary for the occurrence of attribute a .

$$\vec{bf}_a = \forall i \in [1, m] : bf[i] = \max_{j=1}^m H_j(a) \quad (53)$$

$$\min_{i \in [1, m]} \vec{bf}_{load}[i] : \vec{bf}_a[i] = 1 \quad (54)$$

To prevent nodes in the community from linking attributes to nodes via the Bloom filter, the initiator v_0 initializes a the Bloom filter with random noise. Otherwise, node v_1 might assume that \vec{bf}_{load} only contains attributes from v_0 and attempt to determine the attribute combination with a brute force approach. For the randomization, v_0 takes its own number of attributes $|A_{v_0}|$ and multiplies it with $|V'|$ to obtain an estimate of the expect total number of entries in the Bloom filter. Then v_0 adds $|A_{v_0}| \times |V'|$ nonces to \vec{bf}_{load} : draw a random nonce N from the domain of possible attribute pseudonyms, generate a Bloom filter \vec{bf}_N for the nonce, and adds \vec{bf}_N to the Bloom filter containing the load status, i.e., $\vec{bf}'_{load} = \vec{bf}_{load} + \vec{bf}_N$.

$$m_{load} = (\overbrace{\vec{bf}_{load}}^{\text{Bloom filter}}, \overbrace{(t_a, pk_{v_{over}})}^{\text{request}}, \overbrace{(v_{under}, pk_{v_{under}})pk_{v_{over}}}}^{\text{response}}) \quad (55)$$

$$m_{load} = (\vec{bf}_{load}, (0, 0), 0) \quad (56)$$

$$m_{load} = (\vec{bf}_{load}, (t_a, pk_{v_{over}}), 0) \quad (57)$$

The circulated message consist of three elements a given in Equation (55). The first component of the message is the counting Bloom filter itself with $m_{load} = (\vec{bf}_{load})$. In addition, the message is appended with a handover request block. Within that block, the first overloaded node v_{over} specifies the attribute to be handed over via the attribute pseudonym t_a . The empty handover block is denoted by the 0-pseudonym t_0 . Note that the v_{over} does not reveal its ID. However, as v_{over} and the potential underloaded node v_{under} might not be neighbors in the basic overlay G , i.e., $(v_{over}, v_{under}) \notin E$, no secure communication channel between both nodes exists. To overcome this issue, v_{over} generates a new asymmetric key pair $(sk_{v_{over}}, pk_{v_{over}})$ and attaches $pk_{v_{over}}$ to the message as well. Combined, the message becomes $m_{load} = (\vec{bf}_{load}, (t_a, pk_{v_{over}}))$. In order for an underloaded node v_{under} to respond, the message requires a handover response block. This block consists of the ID of v_{under} encrypted with $pk_{v_{over}}$ as well as its own public key $pk_{v_{under}}$. With the response block, the message becomes Equation (55).

When the overloaded node receives a message as given in Equation (55), it can decrypt the handover response with $sk_{v_{over}}$, obtain $(v_{under}, pk_{v_{under}})$, and establish direct communication with v_{under} . Before continuing to circulate the message, it sets the handover request and response to 0. A message without handover request is given in Equation (56). A message with handover request but without response is given in Equation (57).

The system parameter $lf_{over} > 1$ enables nodes to determine if they are overloaded. This parameter is a factor relative to the average load factor lf_{avg} of the community. Equation (58) shows how the average load factor can be estimated based upon the counting Bloom filter the the community size. The constant B determines the average number of 1 bits in a Bloom filter holding one attribute pseudonym. This constant has to be determined given the particularly chosen hash functions for the Bloom filter.

$$lf_{avg} = \frac{\sum_{i \in [1, m]} \vec{bf}_{load}[i]}{B \times |V'|} \quad (58)$$

The following function *overloaded* determines if a node v is overloaded or not. An overloaded node forwards more than lf_{over} times the attributes than the expected

$$overloaded(v, \vec{bf}_{load}, |V'|) = \begin{cases} true & |A_v| \times lf_{over} > lf_{avg} \\ false & otherwise \end{cases}$$

A node with $|A_v| < lf_{avg}$ is an underloaded node.

$$\begin{matrix} & 2 & 1 \\ \begin{matrix} a_1 \\ a_2 \\ a_3 \end{matrix} & \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \end{matrix} \quad (59)$$

$$H_1(a) : \vec{bf}[1] = a[1] \quad (60)$$

$$H_2(a) : \vec{bf}[2] = a[1] \quad (61)$$

$$H_3(a) : \vec{bf}[2] = a[2] \quad (62)$$

$$H_4(a) : \vec{bf}[3] = a[1] \quad (63)$$

$$H_5(a) : \vec{bf}[4] = a[2] \quad (64)$$

Example 20. The matrix (59) shows an example with 3 attributes (rows), and the corresponding bits (columns), which can be used by the Bloom filter hash functions. The lowest bit takes the right most position each. These hash functions are given in a simplified version in Equation (60)-(64) and generate a Bloom filter with $m = 4$ positions. Using these hash functions, the Bloom filters per attribute are the following:

$$\vec{bf}_{a_1} = 0111 \quad (65)$$

$$\vec{bf}_{a_2} = 1010 \quad (66)$$

$$\vec{bf}_{a_3} = 1111 \quad (67)$$

A counting Bloom filter containing each attributes once therefore becomes 2232. If the attribute a_1 is present another time, the counting Bloom filter becomes 2343. With Equation (54) an overloaded node with attribute a_1 would identify positions 1, 3 (right most bit position is 1) as lowest given the Bloom filter \vec{bf}_{a_1} . Correspondingly, attribute a_1 is at most present 3 times in the counting Bloom filter. The overloaded node could therefore assume up to two candidate nodes willing to take over attribute a_1 .

To complete the handover of the forwarder role of an attribute, the node v_{over} has to handover neighbors to v_{under} . The neighbors of v_{over} also have to be informed. Equations (68), (69) remove all overlay neighbors for a from v_{over} . The “prime” versions of the set indicates the status after the handover. Then v_{under} takes over all overlay successors of v_{over} (Equation (70)). Note that v_{under} keeps its existing neighbors, and that v_{under} might be successor of v_{over} and therefore must be removed to avoid cycles. In case v_{under} is successor of v_{over} , it also takes over all predecessors from v_{over} as given in Equation (71). In any case, all predecessors and successors of v_{over} have to be informed as given in Equations (72), (73).

$$N_a^-(v_{over})' = \emptyset \quad (68)$$

$$N_a^+(v_{over})' = \emptyset \quad (69)$$

$$N_a^-(v_{under})' = (N_a^-(v_{under}) \cup N_a^-(v_{over})) \setminus \{v_{under}, v_{over}\} \quad (70)$$

$$N_a^+(v_{under})' = N_a^+(v_{over}) \setminus v_{over} \quad (71)$$

$$\forall v \in N_a^+(v_{over}) : N_a^-(v)' = (N_a^-(v) \setminus v_{over}) \cup v_{under} \quad (72)$$

$$\forall w \in N_a^-(v_{over}) : N_a^+(w)' = (N_a^+(w) \setminus v_{over}) \cup v_{under} \quad (73)$$

The optimization ends when the initiator v_0 observes two empty handover requests in sequence. Any overloaded node, which forwards at least one attribute pseudonym the node has not yet attempted to hand over, would use the empty handover request. If the request is empty when it reaches the initiator again, there must be no such node. The initiator therefore informs all nodes V' to dissolve the community.

The overall overlay optimization, i.e., community formation and optimization process, never terminates. The signaling overhead of the communication via the cyclic community may be considered high— $|V'|$ messages are required from every request to response—, but the cycle also protects the anonymity of all nodes, as attribute pseudonyms cannot be linked to a particular node by the malicious insiders. Two exceptions apply:

1. A malicious insiders claims to be underloaded by responding to every handover request. Then this node can claim traffic by taking over attributes. Furthermore, this node learns which attributes overloaded nodes forwarded before. However, the malicious insider may quickly become overloaded. Moreover, the insider only learns until the other nodes become not overloaded anymore, i.e., only a limited number of attributes can be learned. Furthermore, another node may become suspicious once the same underloaded node answers more than one request.
2. Each underloaded node that takes over an attribute pseudonym learns about this pseudonym and the other node as well.

This section presented an inter-overlay optimization approach based upon a greedy community formation, a ring topology, and counting Bloom filters. Nodes join or create communities repeatedly. Overloaded nodes attempt to find an underloaded nodes within such a community and handover the forwarding role for an attribute pseudonym.

The ring topology in the community works as an anonymization service as no message can be linked to a particular participant—except when an attribute handover request is accepted. The counting Bloom filter ensures that no attribute can be linked to a node. This is particularly relevant as a subscriber will never handover a desired attribute. But the subscriber can still indicate to the community that it is a viable candidate to handover a forwarder role for this attribute to. The random noise added to the Bloom filter ensures that this protection is also achieved during the initial population of the Bloom filter.

The confidentiality remains protected as only attribute pseudonyms and hashes of the same are exchanged.

This overlay optimization scales linear with the number of attributes, like the proposed solutions for the attribute distribution building block. Every node is at most part of one community, and therefore stores at most a list of all its own attributes it attempted to handover (linear). The size of the counting Bloom filter can be also linearly increased with the total number of attributes in the system. The maximum number of nodes in a community remains constant given the neighbor list of all nodes is maintained at constant size by the basic membership management.

This optimization is performed within an optimization community. Such communities are established in a randomized fashion by nodes deciding at a random time to become initiator. The initiator then adds nodes in a greedy manner to the community. These nodes are picked from the neighborhood sets of the initiator, i.e., neighbors within the initiators attribute overlays. This selection is biased towards having multiple nodes per attribute in the community. These communities therefore increase the chances of successful handovers compared to a purely random selection. The initiator connects the community in a ring that communicates via a token.

After the community ring is established, the nodes first establish their joint load status. A token with a counting Bloom filter is circled within the community, and nodes add their attributes to the counting Bloom filter. The counting Bloom filter protects unlinkability as it does not reveal which node contributed which attributes to the counting Bloom filter. Noise is added to the counting Bloom filter before adding the load status to protect malicious insiders from linking attributes to nodes. Nodes within the community can then reason if other nodes handle an undesired attribute as well, add a handover request to the token, and wait for a handover response. In case of a successful response, both nodes establish direct communication to hand over the forwarder role.

4.6 CONTENT DISTRIBUTION

The content distribution delivers publications, or notifications respectively, from publishers to subscribers. These notification messages contain actual content—in contrast to the advertisement, subscription, and overlay optimization messages, which contained at most an attribute pseudonym. Confidentiality is therefore a crucial requirement besides scalability and minimal overhead for this building block. Moreover, authenticity of notifications is required to ensure that these messages have originated by a publisher holding the respective attribute keys.

The system distributes notifications for an attribute a by sending the notification through the respective and pre-established attribute overlay \mathcal{M}_a . To ensure confidentiality, publishers use the shared secret K_a to encrypt the notification content. The resulting notification

is given in Equation (76). Such a notification does not leak more information than the preceding advertisement. Together with the signature scheme from attribute localization the notification becomes the message as given in Equation (77) with $sig = sign(t_a || \{m\}_{K_a}, sk_a)$ where $x||y$ denotes the concatenation of elements x, y .

$$m_{notif} = (t_a, \{m\}_{K_a}) \quad (74)$$

$$m_{notif} = (t_a, \{m\}_{K_a}, sig) \quad (75)$$

A notification for attribute a originated by a publisher p contains pseudonym t_a as routing identifier and some content m . To distribute a notification from a publisher $p \in \mathcal{P}_a$ to all subscribers \mathcal{S}_a , the publisher floods pre-established mesh \mathcal{M}_a . For that, p as well as nodes in \mathcal{M}_a follow Procedure [processNotification](#): first, the node verifies that the signature is valid (lines 1–3). Next, the node obtains the subscription table (line 4) and iterates over all subscription entries (lines 6–13). In case sending node v matches the subscription entry (line 7), the entry is skipped. Otherwise, the node forwards the notification to the node v^* (lines 10–11). The notification is also sent out towards all publishers (lines 14–21), except the direction the notification was received from (lines 16–18). The set *complete* (line 5) maintains a set of nodes the notification has been sent to or received from, such that no duplicates are sent out (lines 11, 20).

Procedure processNotification($(t_a, \{m\}_{K_a}, sig, v), (w, T^{sub}, T^{adv})$)

```

1  if not verify( $t_a, \{m\}_{K_a}, sig$ ) then
2    | return;
3  end
4   $R_{t_a}^{sub} \leftarrow \text{findRecord}(t_a, T^{sub});$ 
5  complete  $\leftarrow \{v\}$ ;
6  for  $v^* \in R_{t_a}^{sub}$  do
7    | if  $v^* \in \text{complete}$  then
8    |   | continue;
9    | end
10   |  $m_{notif} = (t_a, \{m\}_{K_a}, sig);$ 
11   | message( $v^*, m_{notif}$ );
12   | complete  $\leftarrow \text{complete} \cup v^*$ ;
13  end
14   $R_{t_a}^{adv} \leftarrow \text{findRecord}(t_a, T^{adv});$ 
15  for  $(h^*, v^*) \in R_{t_a}^{adv}$  do
16    | if  $v^* \in \text{complete}$  then
17    |   | continue;
18    | end
19    |  $m_{notif} = (t_a, \{m\}_{K_a}, sig);$ 
20    | message( $v^*, m_{notif}$ );
21  end
22  return;
```

While the pseudonym t_a used in advertisements and notifications does not leak plaintext information. For notifications however, the message confidentiality of content m has to be protected as well.

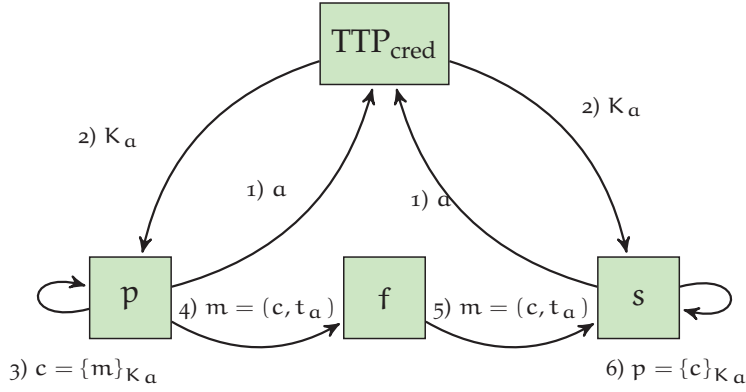


Figure 29: Key exchange and message flow for confidential notifications.

Both participant sets, \mathcal{P}_a and \mathcal{S}_a , share a symmetric key K_a . Publisher p encrypts m using K_a and obtains $\{m\}_{K_a}$. Figure 29 illustrates process to obtain the key: both, p, s request the key K_a for a from the TTP (step 1, 2). The publisher then encrypts the plaintext message m with K_a (set 3), creates a new message $m' = (c, t_a)$, and sends it to f (step 4). The forwarder looks up its routing entry for t_a and forwards m' to s (step 5). The subscriber then decrypts m' and obtains the plaintext message m (step 6). The same signature scheme as for advertisements is applied to obtain an authentic notification as given in Equation (76), with the signature provided in Equation (77).

$$m_{notif} = (t_a, \{m\}_{K_a}, sig) \quad (76)$$

$$sig = sign(t_a || \{m\}_{K_a}, sk_a) \quad (77)$$

The content distribution approach presented in this section protects the anonymity of participants by the same means as presented with attribute localization—attribute pseudonyms and the avoidance of global node IDs. Regarding confidentiality, a notification $m_{notif} = (t_a, \{m\}_{K_a})$ does not contain any more plaintext information than an advertisement. A notification does however leak metadata, such as the message length and the message frequency. This metadata can be exploited by the global observing threat. The following two sections introduce mechanisms to prevent such exploitation. The authenticity of notifications is protected as all nodes, including subscribers, verify the signature of notifications. The scalability of the content distribution depends upon the overlay established by the attribute distribution building block. The overhead arising from forwarding notification is low. No additional storage overhead arises. The processing overhead is limited to a signature verification and a routing table lookup.

4.7 COVER TRAFFIC IN ANONYMOUS PUB-SUB

A global observer threat arises from topological properties—similar to the statistical disclosure attack [33]—of the overlay to identify subscribers and thus breaks subscriber anonymity. This section introduces a novel mechanism called probabilistic forwarding (PF) to mitigate such attacks. Probabilistic forwarding PF does not only introduce cover traffic as the related cover traffic / mimic traffic approaches [11, 57], but increases the size and thus the statistical properties of attribute overlays.

This section is structured as follows: first, the statistical disclosure attack [33] is transferred to anonymous Pub/Sub following the formalization given in Section 3.1 (page 35). Then, probabilistic forwarding is introduced as a countermeasure to mitigate this attack.

STATISTICAL DISCLOSURE ATTACK ON ANONYMOUS PUB/SUB For this attack, the global observer isolates \mathcal{M}_a from the basic overlay G with information obtained from timing or node churn. Given background information in terms of the amount of publishers $|\mathcal{P}_a|$ and subscribers $|\mathcal{S}_a|$, the global observer threat $G\mathfrak{A}$ can set up a probability distribution of all nodes V_a . In case publishers and subscribers dominate the overlay, i.e., $|\mathcal{P}_a \cup \mathcal{S}_a| \div |V_a| > 0.5$, the attacker immediately breaks publisher and subscriber anonymity. That means, the attacker can simply pick *all* overlay nodes as publishers and subscribers for a and is correct in most cases.

The attackers success can be measured via anonymity set sizes (cf. Section 2.2.3, page 18) as classification *accuracy* that is defined as the ratio of correct classifications (true positives TP and true negatives TN) among all classifications (TP, false positives FP, TN, and false negatives FN), Equation 78. Thus, the global attacker succeeds for *accuracy* > 0.5 , i.e., it guesses publishers and subscribers better than a coin flip.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (78)$$

DE-ANONYMIZING LEAF NODES The attacker can extend the introduced attack with timing information. With timing information, the attacker isolates so called *leaf* nodes via message flow from \mathcal{M}_a , i.e., nodes with $N_a^-(v) = \{\}$ that do not forward notifications any further. Hence the attacker may consider such nodes v as subscribers in accordance with the overlay construction.

To analyze overlay meshes, the set specified in Equation (79) denotes nodes that do not send notifications in \mathcal{M}_a . Hence those nodes represent “leaves” in \mathcal{M}_a . The attacker analyzes message flow and isolates $L(\mathcal{M}_a)$, and thus reduces the remaining problem size to $V_a \setminus L(\mathcal{M}_a)$. In case the ratio of publishers and subscribers among those remaining non-leaf nodes is larger than 0.5, i.e., $|\mathcal{P}_a \cup \mathcal{S}_a| \setminus L(\mathcal{M}_a) \div |V_a| > 0.5$, even the entropy of the probability distribution of the remaining problem is too low to protect participant anonymity. There-

fore, \mathcal{M}_a is insufficient to protect against the G2 threat as it does not contain enough cover nodes.

$$L(\mathcal{M}_a) = \{v : v \in V_a, N_a^-(v) = \emptyset\} \quad (79)$$

PROBABILISTIC FORWARDING To mitigate the statistical disclosure attack, this thesis adapts the concepts of cover traffic as introduced by [11] and the so called “mimic traffic” [57] (cf. Section 2.3, page 21) to notifications in a Pub/Sub system. This extension increases the overall number of nodes in \mathcal{M}_a . Moreover, leaf nodes do not indicate subscribers any more and thus mitigate the attacker’s advantage.

Probabilistic forwarding works as follows: upon sending a subscription message, every forwarder and subscriber decides for every neighbor at random if to forward notifications to this neighbor or not. The system parameter $\mu \in [0, 1]$ models a threshold for this decision and therefore expresses the “strength” of probabilistic forwarding (PF). With this process, nodes v pull additional neighbors from $N(v) \setminus \{N_a^-(v) \cup N_a^+(v)\}$ into $N_a^-(v)$. Hence, \mathcal{M}_a grows and thus the chances of randomly picking subscribers decreases. The likelihood of leaf nodes being subscriber decreases as well.

Example 21. Figure 30 exemplifies this concept. Left: subscribers s_1 , s_2 , and s_4 are exposed as leaf nodes. By picking leafs as subscribers the attack accuracy evaluates to $(3 + 1) \div (3 + 0 + 1 + 1) = 0.8$. Middle: subscribers s_1 and s_3 conceal themselves by adding cover neighbors. Thus, the accuracy drops to ≈ 0.43 . Right: cover neighbors may pick additional cover neighbors by themselves—cover node f_3 pulls in another cover node, f_4 (accuracy 0.5).

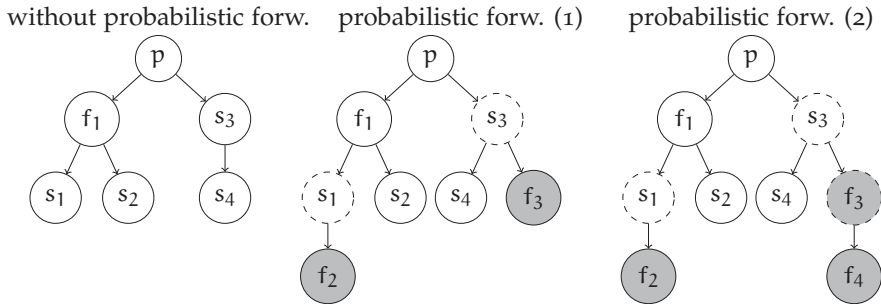


Figure 30: Hiding nodes dashed, receivers of cover messages gray. Middle: probabilistic forwarding with one hop of neighbors. Right: two hops of neighbors.

Formally, subscribers and forwarders $v \in V_a$ determine uniformly at random for every neighbor n as defined in Equation (80) if to forward to this “cover node” or not. The mesh \mathcal{M}'_a is constructed based

upon local knowledge of all participants. Thus, if v selects at least one neighbor, $N_a^-(v) \neq \emptyset$, and v can expect not to be in $L(\mathcal{M}_a)$ anymore.

$$n \in N(v) \setminus N_a^+(v) \setminus N_a^-(v) \quad (80)$$

PF can also be used for multi-hop PF such that PF nodes select additional PF nodes. This behavior is beneficial in cases where less pure forwarders than subscribers are present in the attribute overlay. This behavior prevents malicious insider nodes, which is asked to become a PF node, to distinguish if the requester is another PF node, subscriber, or pure forwarder. Furthermore, PF can be combined with other mechanisms, such as the previously introduced community optimization, and the shell game (SG) which will be introduced in the following section.

SUMMARY In summary, cover messages via probabilistic forwarding (PF) protect leaf subscribers from the global attacker by increasing the size of $anonSet_{adv}$. However, this method depends on the number of cover messages and thus can result in high signaling overhead. Furthermore, this method does change the inner structure of the overlay. The global attacker can therefore still infer information from this structure.

To overcome these drawbacks, this thesis proposes the SG in the following section. The shell game is intended to be used in conjunction with probabilistic forwarding (PF), i.e., with PF nodes, but can also be used independently. In particular, the possibility to combine probabilistic forwarding (PF) with other anonymity protection schemes highlights the advantage of probabilistic forwarding (PF) compared to other cover traffic approaches.

4.8 OVERLAY RANDOMIZATION IN ANONYMOUS PUB-SUB

This section introduces an overlay randomization algorithm called the shell game (SG). The SG restructures the overlay network without exposing to the global observer threat how it is restructured. This is an analogy to the real-world shell game where a conjurer hides a ball under a shell. Then three shells swap their positions. The player wins, if he picks the shell with the ball. However, conjurers cheat the game by placing the ball under another shell, which is hard to observe by the player.

This section is structured as follows: first, the concept of the SG is introduced. Second, the evaluation baseline as well as the formal model for the SG are presented. Third, a formal model for the shell game is introduced. Fourth, the challenge start / stop triggers for the SG is discussed followed by the SG algorithm. Finally, implementation challenges are discussed and a potential anonymity attack is outlined.

SHELL GAME CONCEPT During the content distribution, publishers can send notifications with any message size and at any rate. Considering the global observer threat, such notification flows may be

traceable from the source to the receivers, for instance by linking all notifications of equal size. With such a flow trace, it becomes evident how the overlay was constructed—from the publisher via possible pure forwarders to subscribers. The **SG** follows the goal to randomize the overlay topologies, and therefore the meaning of flow traces of notifications as well. This randomization must be performed after the attribute localization, but before a publisher sends the first notification via the content distribution. Relying on the previously introduced overlay optimization is consequently insufficient. **PF** tackles the same goal, but causes signaling overhead, and most importantly does not randomize the “inner topology” of the overlay. With **PF** only additional nodes are added.

With the **SG**, nodes swap overlay positions by exchanging neighborhood sets. These exchange messages are hidden within heartbeat messages (cf. Section 4.1) and thus prevent the attacker from observing position changes.

Every node starts the **SG** as soon as it has at least one overlay neighbor. As the overlay \mathcal{M}_a starts to form from subscribers towards publishers, \mathcal{M}_a is shuffled before the first notification reaches subscribers. Hence, the global attacker cannot observe \mathcal{M}_a “before” the **SG**. The subscriber anonymity set for the global attacker becomes the size of the whole overlay: $|\text{anonSet}_{adv}(a)| = |\mathcal{V}_a|$. Furthermore, the attacker accuracy for classifying subscribers remains as low as random guessing $\text{accuracy} = |\mathcal{S}_a| \div |\mathcal{V}_a|$.

EVALUATION BASELINE The effectiveness of the **SG** in protecting subscriber anonymity depends on the attack approach as well as the **SG** parametrization. The following approach serves as evaluation baseline: the global observer \mathcal{GA} attempts to separate subscribers from pure forwarders. This attacker is capable of identifying the attribute mesh \mathcal{M}_a by observing the flow of notifications. Furthermore, the attacker has background knowledge about the number of subscribers $|\mathcal{S}_a|$. Given that knowledge, probability of any node $v \in \mathcal{V}_a$ being a subscriber is defined as $P(v \in \mathcal{S}_a) = |\mathcal{S}_a| \div |\mathcal{V}_a|$. If the attacker cannot infer any information from the overlay topology, the chances of the attacker in an optimal **SG** should not significantly deviate from $P(v \in \mathcal{V}_a)$, a random pick.

FORMALIZATION To perform a **SG**, any node $v \in \mathcal{V}_a$ chooses a neighbor $w \in \{N_a^-(v) \cup N_a^+(v)\}$, and sends a message to w asking to swap positions in the overlay. As a result, w hands over $\{N_a^-(w) \cup N_a^+(w)\}$ to v , and notifies these neighbors. Then v completes the **SG** by handing over $\{N_a^-(v) \cup N_a^+(v)\}$ to w and notifies these neighbors as well. Assuming that the basic overlay provides end-to-end connectivity and contains non-subscribing forwarders, this algorithm may result in the positioning of non-subscribers in leaf positions. An attacker should therefore not be able to infer any information from the fact that a node is in a leaf position.

The **SG** is illustrated in Figure 31. This example shows an overlay with one publisher, four subscribers, and three forwarders (left). The

SG is performed two times; the outcomes are shown at the center/-right of Figure 31. The overlay changes in between subsequent SGs are marked in gray; the node that executes the algorithm is dashed.

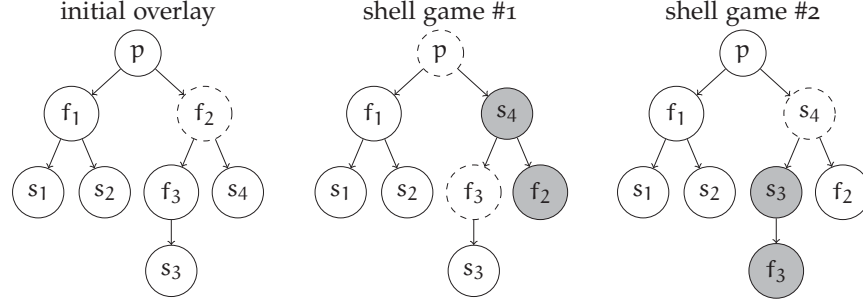


Figure 31: Initial state and two steps of the shell game. The active node is marked with a dashed outline, swapped nodes are marked in gray.

In Figure 31, subscribers s_4 initially joins the overlay, and its parent node f runs the algorithm to swap with s_4 as depicted in the center (SG #1). After that, the nodes p and f_3 obtain new neighbors and also run the algorithm. In step 2, f_3 swaps with child s_3 while p does not swap. Then, s_4 detects another change, new neighbor s_3 , and executes the SG algorithm.

After the start and the termination of the SG have been clarified, the paragraph *Algorithm* will explain the complete SG using pseudocode.

TRIGGER FOR START AND TERMINATION As each node executes the SG independently, a condition is necessary to trigger the start of a SG. Likewise a trigger is required to terminate further SGs. These triggers should prevent unnecessary SG and thus prevent signaling overhead. This thesis uses a subscribing neighbor as start trigger, and exponential decay to slow down the rate of SGs.

Whenever a new subscribing neighbor connects to a node v , i.e., $N_a^-(v)$ grows, v may initiate a SG. That is, whenever S_a grows by a new subscriber, as well as whenever the neighborhood changes due to SGs. Thus, the SG ripples through \mathcal{M}_a permanently. This trigger allows to perform a SG as soon as a neighbor is available. Furthermore, it stimulates the SG when new neighbors become available, e.g., as a result of SGs.

Whenever a new publishing neighbor connects, the very same actions are set in motion. By the procedure of the content distribution, both, the new publisher and the node it connects with, take the role of a subscriber. This ensures that all subscribers receive all notifications from all publishers. Both nodes can therefore immediately attempt to initiate a new SG.

To prevent unnecessary SG, every node that triggered itself for a SG evaluates the exponential decay function (81), randomly draws a number r from the uniform distribution $[0, 1]$, and decides to play the

SG if the result of the functions exceed the random number r . Otherwise, the node waits. The function (81) takes the node's overlay membership time $t(v)$ as parameter. Hence, the probability of this node to play the SG decreases over time. The system parameter λ controls how fast function (81) converges towards 0 and thus the probability of SGs. This approach causes nodes that recently joined the attribute overlay to perform many SGs, i.e., "move quickly" through the overlay, and then become less "lively"

$$P_{\text{swap}}(t) = e^{-\lambda * t} \quad (81)$$

ALGORITHM The pseudocode for the SG is given by Procedure **shellGame**.

Procedure shellGame($N_a^+(v)$, $N_a^-(v)$, $t(v)$)

```

1  $P_{\text{swap}} \leftarrow e^{-\lambda * t(v)}$ ;
2  $r \leftarrow \text{randomUniform}(0,1)$ ;
3 if  $r > P_{\text{swap}}$  then
4    $t_w \leftarrow \text{randomExponential}()$ ;
5   wait( $t_w$ );
6   shellGame( $N_a^+(v)$ ,  $N_a^-(v)$ ,  $t(v)+t_w$ );
7   return;
8  $\text{swapee} \leftarrow \text{pickUniform}(N_a^-(v))$ ;
9 for  $n \in N_a^-(v) \setminus \text{swapee}$  do
10   message( $n$ , new parent: swapee);
11 for  $n \in N_a^+(v)$  do
12   message( $n$ , swap child:  $n$  by swapee);
13  $N_a^-(v) \leftarrow \text{message}(\text{swapee}, \text{new children: } N_a^-(v) \setminus \text{swapee})$ ;
14 return;
```

Each node v maintains its own membership time in the overlay $t(v)$ in addition to neighborhood sets. Upon invocation of this procedure, v rolls a dice with outcome r , compares r with equation (81) and thus either continues with the SG for $r \leq P_{\text{swap}}(t)$. Otherwise, the node draws a waiting time t_w uniformly at random and initiates the SG afterwards again. Then v picks a swapee n from its subscribed neighbors $N_a^-(v)$ and notifies the other neighbors about the swap. That is, all subscribing neighbors, except the swapee, replace their parent v with the swapee, all publishing neighbors replace child v with the swapee, and v exchanges subscribing neighbors with the swapee. Swapper and swapee act as proxies to establish the new confidential and authentic connections to their new overlay neighbors. Thus, nodes must send control messages over authentic and confidential connections.

COLLISIONS & MESSAGE PADDING Concurrent executing of several SGs in the same overlay may affect intersecting sets of nodes and could result in a partitioned overlay. The two-phase commit protocol

[10, 64] resolves this issue by locking adjacent nodes before starting the SG. Multiple SG initiators may attempt to lock the same swapee or adjacent node simultaneously. To overcome this conflict, a node may decline a lock. A timeout t_w forces the initiator to always release a lock.

The size of the messages exchanged during the SG may leak details to the global observer. E.g., a large message could indicate the transport of a neighbor list. To prevent such information leakage, every SG message must be encrypted to ensure the confidentiality of the SG. Moreover, SG messages must be padded to a fixed length; opposed to for instance advertisements, SG messages contain neighborhood sets of variable length. For this, the method of using heartbeat messages has been introduced in the basic membership building block. The SG now dominates the length requirement for these messages.

In detail, assuming 73 bytes size of an advertisement, the SG can hand over $\lfloor (73 - 20 - 1)/4 \rfloor = 13$ IPv4 neighbors at once assuming 20 bytes for the hash chain element, 1 byte for the message type, and 4 bytes per IPv4 address. Thus, SG messages can be indeed embedded the same way as messages for community management under the assumption of a bounded neighborhood size. In dependence of the basic membership, the boundary for the neighborhood is known upfront. A message fragmentation is possible as well, but slows down the SG given the fixed message rate of heartbeats.

As mentioned in the motivation, the SG should be performed before the first notifications are sent out. Likewise, the SG should be performed after new nodes joined but before another notification is sent. Otherwise, the newly joined node(s) and their potential role(s) would be leaked to the global observer threat. To overcome this issue, publishers may indicate pauses in their last notification, allowing overlay nodes to perform SGs before the next notification is sent.

4.8.1 Cornering nodes with the shell game

The SG also introduces an additional attack surface for malicious insiders. The I \mathcal{A} could use the SG to corner a node in a leaf position, then disconnect this node, and let the global observer monitor the following behavior of the disconnected node: a subscriber has an interest in re-connecting to obtain notifications, a forwarder may remain disconnected.

To perform this attack, the malicious insider first has to “corner” the target node, then disconnect it from the overlay. Figure 32 illustrates this approach. Figure 32a shows the initial setup. To analyze s , the malicious insider I \mathcal{A} first has to get close to s . For that, I \mathcal{A} uses the SG. Figure 32b shows that I \mathcal{A} has trapped s in a corner and interrupts the overlay connection. Figure 32c shows now the G \mathcal{A} just need to observe the connection between f and s . If f starts to relay notifications to s , it can be sure that s is a subscriber.

In summary, de-anonymizing subscribers with cornering and disconnecting them is a time consuming approach and might even be

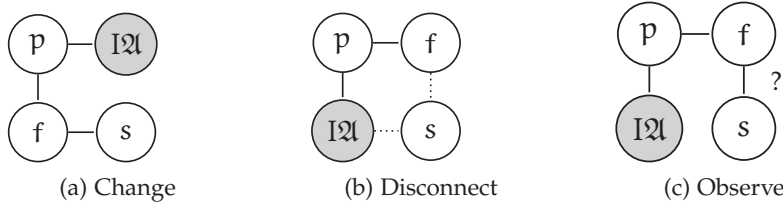


Figure 32: Corner attack: A changes into a position close to s , disconnects s , and then observes s .

detected by genuine nodes. To overcome this attack, all nodes of an attribute overlays should attempt to re-connect after connection failures, even forwarders. In any case, while this attack appears to be a major threat to anonymity, a detailed analysis in the next chapter indicates severe limitations for this attack.

SUMMARY The **SG** uses the basic overlay to create a shuffled unstructured overlay. This works by successively switching the positions of two adjacent nodes. The algorithm does not require a priori knowledge about publishers and subscribers. Compared to probabilistic forwarding, this approach does not depend on additional forwarders. Furthermore, the **SG** already randomizes the overlay during its creation, making it harder to for **G2I** to perform timing-based attacks.

4.9 SUMMARY

This chapter introduced a novel anonymous **Pub/Sub** system called **AnonPubSub** realized via **P2P** overlays. The system is structured according to the building blocks introduced in the previous chapter. Several contributions were made throughout these building blocks, namely: anonymous distribution of attribute knowledge, anonymous inter-overlay optimization, probabilistic forwarding, and the shell game.

KEY MANAGEMENT The presented system uses offline **TTPs** for key management. The **TTPs** are offline as they are only required initially by participants to obtain key material. The first **TTP** is the anonymization **TTP** and authenticates participants. Participants then connect to the second **TTP** to obtain attribute key material. With the keys, publishers can sign and encrypt messages: subscribers can decrypt messages, and all nodes can verify the authenticity / integrity of messages.

ATTRIBUTE LOCALIZATION For attribute localization, this thesis proposed three variations: flooding, forest fire with random walks, and double random walks. These three variations are suited to distribute attribute knowledge with variable node density. All three approaches use hash chains as a combination of transaction pseudonym and distance metric. Nodes can distinguish multiple publishers for the same attribute, and select the shortest path in case of duplicate

messages. Furthermore, hash chains prevent malicious insiders from lying to be closer to the publisher than their predecessors. A novel [TTL](#) counter for forest fire and random walks was introduced that prevents long paths while also preventing anonymity attackers from inferring the precise distance to the publisher or subscriber.

COMMUNITY MANAGEMENT The presented attribute localization mechanisms do not guarantee an equal distribution of load among nodes. Furthermore, node churn may also cause unequal load distribution. To overcome this issue, a novel mechanism for anonymous inter-overlay optimization was presented for community management. The mechanism establishes random ad-hoc optimization communities in greedy manner. The communities exchange load information via counting Bloom filters in such a way that attributes cannot be linked to nodes. According to the load information, overloaded nodes hand over attributes to underloaded nodes in a privacy-preserving manner.

CONTENT DISTRIBUTION For content distribution, two novel mechanisms to protect anonymity were presented. The first mechanism, probabilistic forwarding, is inspired by mimic traffic and prevents global observers from linking nodes and de-anonymizing subscribers. The second mechanism, the shell game, shuffles attribute overlays in such a way that the global observer cannot observe the shuffling. The shell game only causes low signaling overhead and prevents the global observer as well as the malicious insider from inferring information from the overlay topology, e.g., to de-anonymize nodes.

In summary, AnonPubSub constructs and optimizes attribute overlays. Several constructions allow to minimize the signaling overhead given a specific scenario. The inter-overlay optimization compensates negative effects of node churn and distribute load equally. The overlay construction in combination with probabilistic forwarding and the shell game ensures that the subscriber anonymity is protected against the malicious insider threat as well as the global observer threat.

EVALUATION

This chapter evaluates AnonPubSub with respect to both, the anonymity requirements and the signaling overhead. The evaluation exposes the system to the global observer threat $\mathcal{G}\mathcal{A}$ and malicious insider threat $\mathcal{I}\mathcal{A}$. For that, this chapter uses three types of evaluation: first, a qualitative discussion is performed in Section 5.2. Second, an extensive quantitative simulation is performed with the OMNeT++ discrete event simulator in Section 5.3. Third, a prototype is presented as an empirical evaluation in Section 5.4.

Attack approaches used by global observers have been studied extensively [21, 44, 45, 151], while literature approaches that use malicious insiders are more scarce (cf. Section 2.3.6.2, page 23). This chapter therefore first introduces a novel request / response-based attack for malicious insiders in Section 5.1, which will be included in the evaluation in the consecutive sections as well.

5.1 REQUEST / RESPONSE-BASED INTERNAL ATTACK

This thesis introduces a novel attack on anonymous communication systems in Section 3.4.1, page 45. This attack uses timing information as well as request / response protocol semantics found in (anonymous) P2P as well as Pub/Sub systems. This attack can be also applied to AnonPubSub and should be therefore considered for this evaluation. The following paragraphs explain how the attack can be performed to break subscriber anonymity followed by an algorithmic implementation of the attack as published in [38].

ATTACK SKETCH FOR PUB/SUB The attack procedure assumes that the attacker, the combination of $\mathcal{G}\mathcal{A}$ and $\mathcal{I}\mathcal{A}$, has obtained background knowledge of the network topology and the average delay δ_{avg} per hop, and can exploit protocol features to perform a timing attack. For that, the nodes $v \in C_a$ then use a subset $N(v)' \subseteq N(v)$ of their neighborhood for this attack. For every neighbor in $N(v)'$, the attacker first un-advertises the target attribute a and then sends a new advertisement m_{adv} to a selected neighbor. The attacker waits for possible responses in form of subscriptions. The received subscription messages m_{sub} are then used to calculate the distance to the subscriber. Using this distance, the attacker is able to compute three subsets of nodes:

1. First, the subset of nodes closer than the distance of the responding subscriber. These nodes are excludable since they can not be subscribers.
2. Second, the set containing the nodes in the recognized distance. These nodes have an equal probability of being subscribers.

3. The third set computed by the attacker contains all nodes further away than the recognized subscriber. As these nodes are located “behind” subscribers, the attacker can not make any strict assessments regarding their membership in \mathcal{S}_a .

Using these information, the attacker updates its probability map \mathbf{v} in every step (cf. Section 3.4.1). This map associates every node v with its probability of being a subscriber. The attacker considers every node that exceeds the threshold as subscriber, i.e., $v_x \in \mathcal{S}'_a : \mathbf{v}_s[v_x] > \mathbf{v}_0[v_x]$. The following paragraph elaborates on how this can be realized algorithmically.

ATTACK ALGORITHM The Algorithm 7 formalizes the attack sketch. In line 2, the attacker iterates over its neighbors to perform the attack. More neighbors should result in higher attack chances. Likewise, with multiple malicious insiders, the pseudocode is executed sequentially for each insider but with one shared probability distribution. Line 3 dissolves the attribute overlay \mathcal{M}_a in the neighborhood of the attacker. In line 4 an advertisement is sent. If a response, m_{sub} is received in line 6, the attacker estimates the distance [hops] to the subscriber in line 7. The variables k and l in lines 8,9 indicate the number of potential nodes at or further than distance d . To estimate that, background knowledge of G from the global observer is required. For every malicious insider, the attacker uses a tree view upon G with the malicious insider (c) as root. This tree is constructed via a pairwise shortest path search from c to every node G and the hop count as distance metric.

Procedure attack(G, c)

```

1 // Initialization
2 foreach  $v \in N(c)$  do
3   send( $m_{unadv}, c, v$ )
4   send( $m_{adv}, c, v$ )
5    $d \leftarrow \infty$ 
6   if response_received then
7      $d \leftarrow \frac{\delta(m_{adv}, m_{sub})}{2 \cdot \delta_{avg}}$ 
8    $k \leftarrow \text{candidatesAt}(d)$ 
9    $l \leftarrow \text{candidatesFurther}(d)$ 
10  foreach  $u \in \text{branch}(v)$  do
11     $d' \leftarrow |\text{path}(c, u)|$ 
12    if  $d' < d$  then
13       $\mathbf{v}[u] \leftarrow 0$ 
14    else if  $d' = d$  then
15       $\mathbf{v}[u] \leftarrow \lambda \cdot \frac{|\mathcal{S}'_a|}{(k+l)}$ 
16    else if  $d' > d$  then
17       $\mathbf{v}[u] \leftarrow \phi \cdot \frac{|\mathcal{S}'_a|}{(k+l)}$ 

```

EXAMPLE Figure 33a provides an example with malicious node v_0 in the center and all other nodes position by distance in concentric circles. The network without attacker has $|V| = 9$ nodes and $|\mathcal{S}_a| = 2$ subscribers.

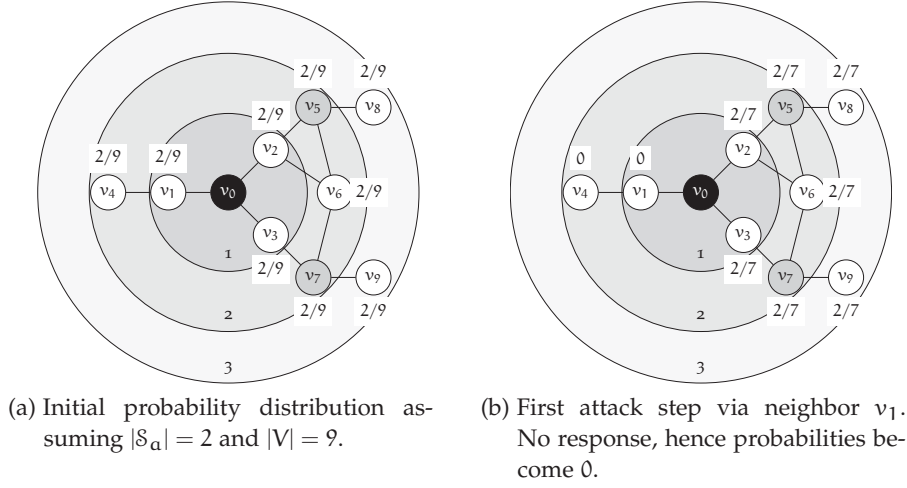


Figure 33: First two attack steps.

The attacker first sends m_{adv}^0 via v_0 to neighbor v_1 , but does not receive any response (Algorithm 7, lines 1-4). Thus, the attacker sets the values of v_1 and v_4 in \mathbf{v}_0 to zero (lines 11-12) and builds a more precise probability distribution \mathbf{v}_1 as shown in Figure 33b, i.e., $j = 2$, $k = 0$, $l = 7$. In the second step, the attacker sends m_{adv}^0 from v_0 to neighbor v_2 and observes $\delta(m_{adv}^0, m_{sub}^0) = 4 \times \delta_{avg}$ (lines 4-5). Hence, the closest subscriber must be two hops away and the attacker adjusts \mathbf{v}_1 to \mathbf{v}_2 as shown in Figure 34a, i.e., $j = 3$, $k = 2$, $l = 4$.

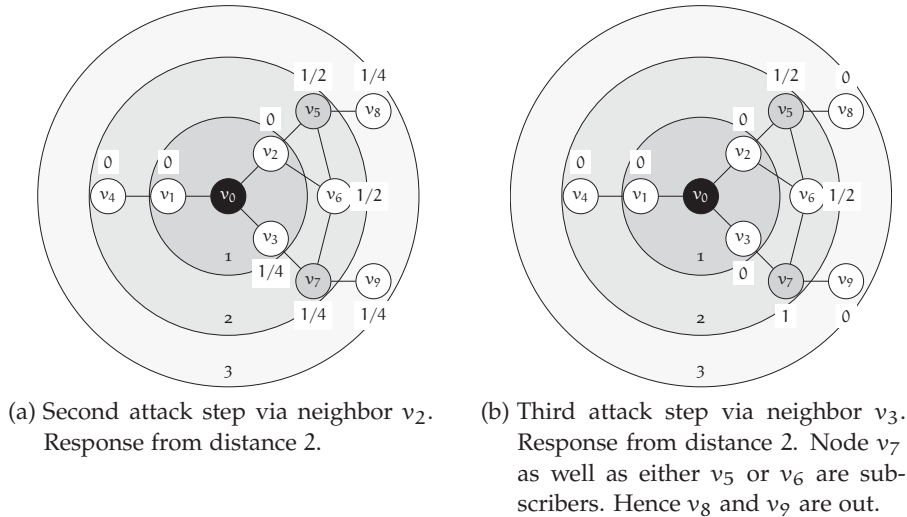


Figure 34: Third and fourth attack step.

In this third and last step the attacker performs two operations: first, he sends m_{unad}^0 to v_2 as he received his m_{adv}^0 via v_3 again—hence both nodes are connected and therefore v_3 already knows the advertised topic and would discard the new advertisement for this topic.

Second, the attacker sends m_{adv}^0 to neighbor v_3 and again observes $\delta(m_{adv}^0, m_{sub}^0) = 4 \times \delta_{avg}$. The only node within the range of 2 hops must be v_7 and is therefore a subscriber. Hence the attacker adjusts the values of v_3, v_7 , and v_9 in \mathbf{v}_2 to 0 and 1, respectively, as shown in Figure 34b, i.e., $j = 3$, $k = 3$, and $l = 2$.

In summary, this example illustrates an easy case for the attacker. The attacker successfully breaks the anonymity of subscriber v_7 , rules out many non-subscribers (TN!s (TN!s)), and suspects v_5 with a high probability— $1/2$ compared to the a priori guess of $2/9$. The following section will evaluate this attack as well as PF and SG in detail via simulation.

5.2 QUALITATIVE PROPERTIES

This section discusses if and how AnonPubSub fulfills the Pub/Sub security requirements as introduced in Section 2.1.2 (page 11) and the anonymity requirements as specified in Section 3.2, page 39.

CONFIDENTIALITY AnonPubSub provides *notification confidentiality* and *subscription confidentiality* as those messages only contain encrypted and pseudonymous information. The function to derive attribute pseudonyms also depends upon knowledge of a secret key. The pseudonyms contained in advertisements and subscriptions therefore do not reveal the attribute to the G2 and forwarders. The secret key is shared among publishers and subscribers related to an attribute. Furthermore, the TTP has access to this key. The security of the key therefore depends upon the safekeeping of these participants.

INTEGRITY The system enforces integrity and authenticity via signatures of advertisement and notification messages to prevent local attackers from spamming attributes and notifications. Subscriptions however do not require authenticity as they do not threaten scalability—duplicates subscriptions can be dropped immediately.

AVAILABILITY (RESILIENCE) AnonPubSub can detect node failure via heartbeats and recover via the repair mechanism introduced as part of community management. The next section will analyze the effect of churn in detail. In any case, the system only maintains only one connection towards publishers each, i.e., it does not provide redundancy.

AUTHENTICATION Authentication applies to sender authenticity in order to obtain key material as well as to message authenticity. Nodes initially connect with the anonymization TTP. The TTP can enforce authentication before establishing a connection to the credential TTP. Message authentication is provided for advertisements and notifications. Every forwarder can verify the authenticity of such a message, but cannot identify a particular publisher. Malicious insid-

ers without key material therefore cannot create false advertisements and overload the system (processing overhead and storage overhead).

ACCOUNTABILITY Accountability of participants to particular messages is not considered here in favour of anonymity. That means, messages cannot be linked with individual nodes.

ANONYMITY AnonPubSub ensures subscriber anonymity against the non-colluding $\mathcal{I}\mathcal{A}$ with anonymity sets $\text{anonSet}_{adv} \geq |\mathcal{N}_a^-(v)| + 1$. For publisher anonymity, the malicious insider possesses key material and imposes a subscriber. The attacker constructs a tree in G , rooted by himself, which contains the shortest paths to all nodes. The attacker then removes branches from which he does not receive advertisements on the shortest path. Thus, the remaining branches contain at least one publisher each, and the size of each branch forms the anonymity set anonSet_{adv} for the contained publisher. The publisher can estimate anonSet_{adv} to be at least of size $|\mathcal{N}_a^+(v)| + 1$ —his predecessors in the mesh plus himself.

Likewise for subscriber anonymity, the malicious insider imposes a publisher. The attacker constructs a shortest path tree in G as well, advertises an attribute, and then removes all branches that do not respond with a subscription. Subscribers can estimate their minimal anonymity sets with $|\mathcal{N}_a^-(v)| + 1$.

PF and the **SG** extend this anonymity protection to the $G\mathcal{A}$ as well. Both mechanisms obfuscate attribute overlay in such a way that attack schemes performed by the $G\mathcal{A}$ become ineffective. In combination, the anonymity is protected against the combination of $\mathcal{I}\mathcal{A}$ and $G\mathcal{A}$. The next sections elaborate on this attacker models in detail with the help of a quantitative evaluation.

The six building blocks are specifically designed to minimize the surface for anonymity attacks. Attribute localization omits global node **IDs** and therefore maximizes anonymity sets. Attribute localization only depends on pseudonyms and the direct neighborhood of a node to prevent the exposure of participants. Likewise, the mechanisms presented for community management also only depend on the local neighborhood and pseudonyms as well as probabilistic data structures. That limits the information an $\mathcal{I}\mathcal{A}$ can obtain and also suppressed accurate reasoning over the few obtained information. The content distribution is limited to the same data structures and information as attribute localization. Moreover, the overlay obfuscation achieved by the community management prevents the $G\mathcal{A}$ from anonymizing subscribers, independent of message rate and delays. The key management introduces an anonymization step to prevent the association of key and therefore attribute with participants, event against a malicious **TTP**.

In summary, AnonPubSub protects publisher anonymity against the $\mathcal{I}\mathcal{A}$ and publisher as well as subscriber anonymity against the $G\mathcal{A}$ as well. AnonPubSub protects anonymity against $\mathcal{I}\mathcal{A}$ for an anonymity set size k that can be estimated by the number of neighbors by each

node v : $k \geq |N_a^-(v)| + 1$. The next section analyzes the attacker $G\mathfrak{A}$ and $I\mathfrak{A}$ via simulation and determines parameters for PF and the SG .

SCALABILITY AND LOW OVERHEAD In terms of memory overhead, AnonPubSub scales proportionally with the number of attributes and neighbors. In terms of communication overhead, advertisements cause the most signaling costs by flooding the basic overlay per attribute. The signaling costs for subscriptions and notifications are bounded by $diameter(G) \times |S_a|$ where $diameter(G)$ denoted the diameter of the basic overlay. The distribution of attribute knowledge via random walks and forest fire further reduce the signaling overhead.

PF and the SG cause additional signaling costs. Each shell game between two nodes v_1, v_2 requires the amount of messages indicated in equation (82) including the two-phase-commit protocol. This equation contains the messages both swapping nodes have to exchange with their respective overlay predecessors and successors.

The overlay diameter reaches its maximum with $|N_a^+(v_x)| \approx 1$. The total message overhead can be therefore approximated with $3 \times |N_a^-(v_1)| + |N_a^-(v_2)| + 4$ messages for this case. The actual signaling costs of the SG as well as PF highly depend on the basic overlay topology and have to be analyzed further via simulation in Section 5.3.

$$msgs = 3 \times |N_a^+(v_1)| + |N_a^+(v_2)| + 3 \times |N_a^-(v_1)| + |N_a^-(v_2)| \quad (82)$$

5.3 QUANTITATIVE SIMULATION

This section analyzes AnonPubSub in detail via a realistic network-based simulation using IPv4 and UDP . For that, AnonPubSub is implemented as application using UDP sockets, which requires the application for instance to comply with the UDP maximum transmission unit. This quantitative simulation will provide insights into the parametrization of AnonPubSub, PF , as well as the SG . Furthermore, the influence of the basic overlay, e.g., network topologies, and participant distributions are investigated. Most significantly, anonymity attacks have been implemented into a this framework and allow analyzing attacks given varying attacker capabilities.

This section is structured as follows: first, an overview of the simulation model is provided. Second, the evaluation metrics for this model are discussed. Next, AnonPubSub is analyzed with this model starting with the basic membership.

5.3.1 Simulation

The simulation for AnonPubSub is based on the OMNeT++ [115] discrete event simulator in combination with the INET¹ framework as introduced in Section 2.5.2, page 30. INET provides the means to simulate the IPv4 stack including associated protocols such as DHCP,

¹ <http://inet.omnetpp.org>

as well as datalink layer protocols such as Ethernet and PPP. The simulation uses one central router to provide routing in between all nodes. This topology denotes a worst case scenario for anonymity as the G \mathcal{A} can establish the communication relation between all nodes by observing this one router. Every node is connected via an individual link, each having its own data rate and transmission delay. The internal structure of the nodes is shown in Figure 35. Every node is connected via network interfaces to the Internet, having the ISO/OSI layer 3 depicted in the center, and layer 4 implementations like UDP on top. AnonPubSub has been implemented as a UDP application and is anchored at the position of the red marking in Figure 35.

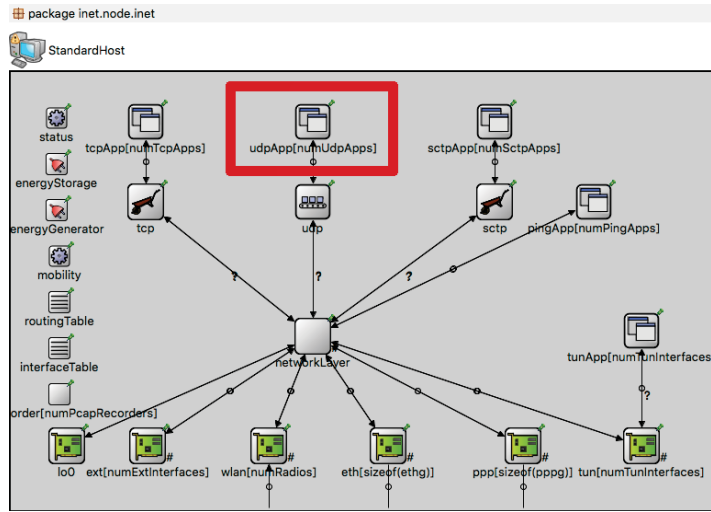


Figure 35: StandardHost structure in the INET framework representing a node $v \in V$.

The selection of UDP allows AnonPubSub to control packet fragmentation internally in order to prevent the unintentional emission of packets such as acknowledgements and fragmented packets to the G \mathcal{A} . The internal structure of AnonPubSub is depicted in Figure 36. The highlighted red connection links to the node just introduced. The central module realizes the forwarder role, the top module the publisher role, and the bottom module the subscriber role. The optional attacker module to the right represents the malicious insider. Various other components such as the G \mathcal{A} have been implemented as well.

5.3.2 Setup

Every node is equipped with an initial neighborhood for basic membership. This initial neighborhood is assigned from a randomly generated topology that follows a given structure. This evaluation uses small-world networks generated according to the Watts-Strogatz model [165], scale-free networks generated according to the power law out degree algorithm [108] with recommended parameters $\alpha = -0.08$ and $\beta = 4.5$, as well as random graphs. The Watts-Strogatz model reflects a social graph as it would be used by darknets [24]. The scale-free models reflects OSNs with few highly connected and many minimal

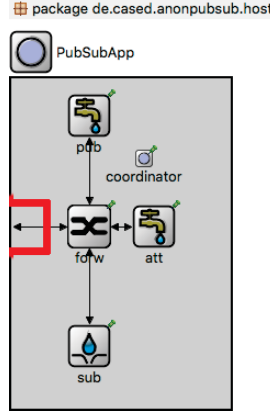


Figure 36: Overview of the AnonPubSub implementation in OMNeT++ as UDP application.

connected nodes. The random graph models an ideal basis for an anonymization systems and is also reflected by P2P systems such as SCAMP aiming to maintain a constant neighborhood size. The resulting basic overlays have an average diameter of 5 – 6. This follows the best practise established by Tor [44], i.e., a sender, a receiver, and 3 guard nodes. For instance for the scale-free model with $|N| = 500$ nodes, this results in an average node degree of $|N(v)| \approx 6.875$. Table 7 summarizes the parameters of the simulation model. The parameter μ models the probability of each node for probabilistic forwarding. The parameter λ controls the exponential decay of the shell game. The parameter $\mathcal{C}_a \subseteq V$ denotes the set of malicious insiders. The parameter δ with a default setting of $\delta = \mathcal{U}(1\text{ms}, 20\text{ms})$ describes the delay distribution of edges in the basic overlay. Here, $\mathcal{U}(1\text{ms}, 20\text{ms})$ represents the uniform distribution within the interval $[1\text{ms}, 20\text{ms}]$. With this distribution, the slowest connection is by a magnitude of 20 slower than the fastest one.

According to the description of AnonPubSub in the previous chapter, the system is simulated with three setups: base line (BL) for the baseline system. PF for probabilistic forwarding, and SG for the shell game.

5.3.3 Evaluation metrics

The following metrics (Table 8) measure the outcome of the experiments: the number of overlay nodes in comparison to all basic membership management nodes, the total signaling costs in messages, the total number of performed swaps (position changes due to the SG), the ratio of leaf subscribers among all leaf nodes within the overlay, and the global attackers accuracy for classifying subscribers, the attacker’s information gain, and the inter-node distance.

The size of the attribute mesh $|V_a|$, and the *accuracy* of the adversaries’ subscriber classification, which is the ratio of correctly classified nodes among the total number of overlay nodes. Lower accuracy therefore means better anonymity. The entropy / information gain (cf. Section 2.2.3, page 18) describes the attacker’s success in

PARAMETER	EXPLANATION
$ V = [100, 1000]$	size of the basic overlay
$ S_a / V \in (0, 1)$	ratio of subscribers among $ V $
$ S_a / V_a \in (0, 1]$	ratio of subscribers among $ V_a $
$ \mathcal{P}_a = 1$	number of publishers per attribute
$ \mathcal{A} = 1$	number of attributes in the system
$publications \in \mathbb{N}$	number of publications from publisher
$ N(v) \in [2, 16]$	node degree in the basic overlay
$\varphi \in [0, 0.2]$	ratio of node churn
$\mu \in [0, 1]$	forwarding probability (PF)
$\lambda \in [0, 1]$	decay parameter (SG)
$ \mathcal{C}_a = [0, 5]$	set of malicious insiders
$\delta = \mathcal{U}(1\text{ms}, 20\text{ms})$	connection delay distribution
$runs = [25, 300]$	repetitions per setup

Table 7: Simulation parameters

METRIC	EXPLANATION
$ V_a \in [1, V]$	size of the attribute mesh
$costs \in \mathbb{N}$	signaling costs in#messages
$swaps \in \mathbb{N}$	no. of swaps performed
$ S_a \cap L(\mathcal{M}_a) / L(\mathcal{M}_a) \in [0, 1]$	ratio of leaf subscribers in \mathcal{M}_a
$accuracy \in [0, 1]$	subscriber classification accuracy
$g \in [0, 1]$	information gain

Table 8: Simulation metrics

breaking anonymity incrementally over time. The information *gain* is calculated as entropy delta between an attack step and the initial a priori entropy. The gain represents the overall success of the attack. The next paragraph describes the information gain in detail.

The arising signaling overhead, e.g., with PF, is measured via *costs* and describes the total number of messages sent. The ratio of leaf subscribers among the total number of overlay leaf nodes tells whether the overlay topology exposes information to G \mathfrak{A} , i.e., a higher ratio of leaf subscribers than overlay subscribers, or not. The number of published messages is collected as a time and overlay invariant measure of the system age.

INFORMATION GAIN The attacker gains information in de-anonymizing members of the set S_a during his attack against subscriber anonymity. To assess the effectiveness of the attack manifold metrics can be used (cf. Section 2.2.3, page 18). This evaluation is based upon sizes and information gain, a metric derived from entropy. These met-

rics are explained in the following with respect to the system model at hand.

The set size is the first metric. For that, the relation of the estimated size of the subscriber set $|\mathcal{S}'_a|$ and the real size $|\mathcal{S}_a|$ has to be calculated. The ratio $|\mathcal{S}'_a|/|\mathcal{S}_a|$ represents a metric similar to the *k-anonymity* [150]. The attacker attempts to reduce the candidate set \mathcal{S}'_a towards the real subscriber set \mathcal{S}_a . Obviously, \mathcal{A} can exclude \mathcal{C}_a , so \mathcal{S}'_a is initialized with $V_a \setminus \mathcal{C}_a$. A high value shows the inefficiency of the attack, the members of \mathcal{S}_a are hidden in a large *anonymity set*, the overlay.

As the first metric is coarse and does not allow a detailed statement about the membership to \mathcal{S}_a per node in the overlay, a probability distribution to describe the information gain of the attacker in more detail has to be constructed. The probability distribution contains the probability of each node in V_a being part of \mathcal{S}_a from the perspective of the attacker. The probability of a single node $v \in V_a$ of being part of \mathcal{S}_a is described as follows: The probability is 0 if $v \in \mathcal{C}_a$, as the node obviously can be excluded since it is controlled by the attacker itself, if $v \notin \mathcal{C}_a$, the probability is the $|\mathcal{S}_a|$ divided by the number of remaining candidates $|V_a| - |\mathcal{C}_a|$. This is the a priori assumption of the attacker. Using this probability, the sum of all probabilities is equal to the number of subscribers. To build the probability distribution, the probability of each node by dividing it by $|\mathcal{S}_a|$ is normalized. For each node in V_a , except the adverse ones \mathcal{C}_a , the same initial probability is used to compose a vector \mathbf{v} holding the probability for every node of being a subscriber to a as given in Equation (83).

$$\forall v \in V_a : \mathbf{v}[v] = \begin{cases} 0 & \text{if } v \in \mathcal{C}_a \\ \frac{|\mathcal{S}_a|}{(|V_a| - |\mathcal{C}_a|) * |\mathcal{S}_a|} & \text{else} \end{cases} \quad (83)$$

$$\sum_{v_i \in V_a} \mathbf{v}[v_i] = 1 \quad (84)$$

Constraint (84) ensures that the probabilities remain valid. The attacker adjusts the map \mathbf{v}_s in every attack step s . The entropy H_s is calculated to quantify the information gain g_s of the attacker in every attack step. Equation (86) defines the entropy based upon the difference between the initial and the current probability per node (Equation (85)). The information gain g_s is defined as difference of the entropy of the preceding attack step $s - 1$ and the entropy of the current attack step s , shown in Equation (87).

$$p_{\text{diff}}(v_i, s) = 1 - |\mathbf{v}_0[v_i] - \mathbf{v}_s[v_i]| \quad (85)$$

$$H_s = - \sum_{v_i \in |V_a|} p_{\text{diff}}(v_i, s) * \log_2(p_{\text{diff}}(v_i, s)) \quad (86)$$

$$g_s = H_{s-1} - H_s \quad (87)$$

Hence, an attacker attempts to minimize H by maximizing g_s in each step, which can be accomplished by eliminating nodes from \mathbf{v}_s

(setting their probability value to 0). The elimination of nodes is possible if the attacker can estimate the position of subscribers in the overlay \mathcal{M}_a and therefore exclude nodes from being members of \mathcal{S}_a .

5.3.4 Basic overlay properties and attack accuracy

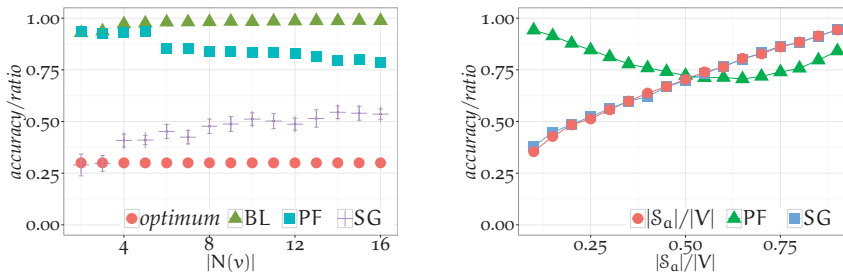
This section analyzes the influence of the basic overlay, the overlay constructed by the basic membership building blocks. In particular, this section addresses the following research questions:

1. Does the neighborhood of the basic overlay influence the anonymity, considering the global observer threat?
2. What is the influence of the number of subscribers per attribute on the anonymity?

To analyze the influence of the node degree on the attacker's subscriber classification accuracy, the node degree of the basic overlay is varied and simulated each with *BL*, *PF*, and *SG*.

The subscriber ratio $|\mathcal{S}_a|/|V| = 0.1$ is fixed, and the average node degree is varied with $|N(v)| \in [2, 16]$. For *PF*, the forwarding probability is set to $\mu = 0.7$. For *SG*, the decay factor is set to $\lambda = 0$ (no decay). All metrics are measured after 500 swaps. A preliminary simulation study indicates that this subscriber ratio creates overlay networks that span across the whole diameter of the basic overlay. Furthermore, $\mu = 0.7$ appears to balance anonymity and signaling overhead.

For *BL*, the classification accuracy is expected to increase with node degree. A higher degree leads to a lower diameter of G and therefore shorter paths in the attribute mesh, containing less forwarders. For *PF*, an overall lower accuracy is expected. Probabilistic forwarding should benefit from more neighbors and therefore result in lower classification accuracy with higher node degree.



(a) BL/PF/SG. Attack accuracy in dependence on the node degree for BL, PF, and SG in comparison to the accuracy of optimal anonymity. (b) PF/SG. Attack accuracy in dependence on $|\mathcal{S}_a|/|V|$ for PF and SG.

Figure 37: Evaluation of the anonymity attack accuracy.

Figure 37a shows the accuracy for the three experiments. The attacker accurately picks subscriber in AnonPubSub without protective

measures (BL). The attacker achieves a mean accuracy of 0.927 with a node degree of 2, and reaches 0.988 with a node degree of 16 (BL). Thus, BL is vulnerable against the global attacker. PF does not only compensate higher node degree, but also benefit from the number of available neighbors. Therefore, PF can improve anonymity by lowering the attack accuracy (0.787 for a node degree of 16). SG provides the best anonymity protection in the scenario (0.29 for a node degree of 2), but the protection degrades with higher node degree due to the lack of forwarder in the attribute mesh (0.536 for a node degree of 16). As reference, $|S_a|/|V_a| = 0.299$ determines the chances of randomly picking subscribers from overlay nodes.

The ratio $|S_a|/|V|$ influences the size of the overlay as well as the ratio of subscribers to forwarders in the attribute mesh. A varying subscriber ratio $|S_a|/|V|$ is used to evaluate its influence on the accuracy of the G \mathfrak{A} when PF or the SG are applied.

The same parameters as in the experiment before are used, but with a larger average neighborhood of $|N(v)| = 4$. The simulation is performed with $|S_a|/|V| \in [0.1, 0.9]$ and the classification accuracy of the G \mathfrak{A} is measured each after 500 swaps.

With increasing $|S_a|/|V|$, more subscribers are expected to take the forwarder role as well. Hence, the classification accuracy for PF should drop. However, for very high ratio, the availability of nodes not part of the attribute mesh shrinks, and thus PF should not be able to conceal subscribers any more. For SG, low ratios should lead to the lowest attack accuracy as many forwarders in the overlay can be used to swap.

Figure 37b shows the attack accuracy with both variations. The accuracy for PF reaches its lowest point with $|S_a|/|V| = 0.65$. Here, $\mu = 0.7$ leads to the best mixture of subscribers and publishers within the overlay. Higher ratios lead to higher attack accuracy again, as expected. SG benefits most from low ratios of subscribers as this leads to the lowest attack accuracy.

In summary, while it appears the PF delivers better anonymity protection than the SG for high subscriber ratios such as $|S_a|/|V| = 0.65$, it has to be noted that such a scenario is hard to achieve. This scenario would indicate that the majority of the participants using the system are interested in one particular attribute. In this case, the fact that a participant is using the system is statistically already sufficient for de-anonymization. In comparison, the SG performs very for low subscriber ratios.

SUMMARY To summarize the research questions, the neighborhood does indeed influence the anonymity. A bigger neighborhood negatively influences the anonymity for the SG, whereas the anonymity for PF improves. Regarding the number of subscribers, the anonymity with PF improves with the subscriber ratio, but only up to a turning point of about 67% subscribers. The anonymity with the SG degrades with increasing subscriber ratio.

5.3.5 Forwarding probability and attack accuracy

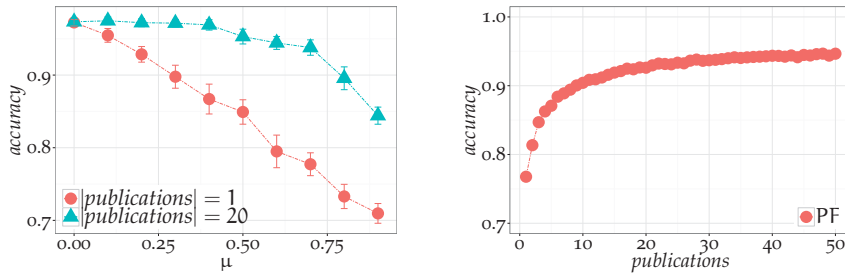
This section analyzes the mechanism PF with the following research questions:

1. How does the forwarding probability μ of PF influence the anonymity?
2. What is the influence of PF on the signaling overhead?

The forwarding probability of $\mu = 0.7$ has been used in the previous experiment to analyze the impact of the basic membership management. This experiment analyzes the influence of μ on the attack accuracy and the signaling costs to identify good values of μ .

Compared to the previous experiment, the subscriber ratio is fixed $|\mathcal{S}_a|/|V| = 0.1$ and PF is simulated with $\mu \in [0.1, 0.9]$. Throughout the experiment, the attack accuracy, the induced signaling costs of cover messages, and complete overlay size are measured to determine if PF exhaust available nodes.

The attack accuracy should decrease proportionally with increasing μ as this corresponds to a ratio of μ leaf nodes covering themselves with PF. Likewise, the costs due to signaling overhead should increase.



(a) PF. Attack accuracy in dependence on μ , each after $\text{publications} = 1$ and $\text{publications} = 20$. (b) PF. Attack accuracy in dependence on publications for PF.

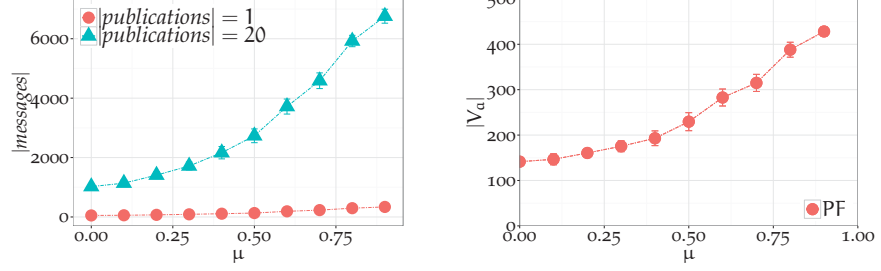
Figure 38: Influence of PF on the anonymity attack by the G2L.

Figure 38a shows the decreasing attack accuracy over increasing μ . With only one publication from the publisher, the accuracy drops proportionally as expected. However, with 20 publications (blue triangles), altering neighbor selection during PF, e.g., due to node churn in the basic membership management, neglect the benefits of PF.

In order to explain this effect, another simulation with $\mu = 0.7$ and $|\mathcal{S}_a|/|V| = 0.3$ is performed. Figure 38b shows an increase in attack accuracy with every publication. Thus, nodes of the attribute mesh should ensure that an attacker cannot intersect candidate sets over time (here publications), e.g., by selecting the same neighbor if possible or even leaving the attribute mesh.

Figure 39a shows the signaling costs in our system. With one publication, the number of cover messages seems to increase proportionally as expected. However, with 20 publications, the cover messages increase polynomially. A leaf subscriber, in a basic overlay network

with node degree 4, may forward probabilistically to up to 3 neighbors, e.g., to replicate and maintain topological properties of the attribute, instead of only one. In any case, the anonymity optimum of $accuracy = 0.5$ is not being reached, and neither reach worst case costs of $(|V| - 21) = 479$ for one publication with 21 subscribers, and $479 * 20 = 9580$ for 20 publications via broadcast each.



(a) PF. Number of messages sent in dependence on μ , each after $publications = 1$ and $publications = 20$. (b) PF. Overlay size in nodes over μ for PF.

Figure 39: Overhead caused by PF.

Figure 39b explains this increase in signaling costs with respect to the overlay size. That is forwarders, subscribers, and cover nodes. Compared to BL ($\mu = 0$) with an attribute mesh of 141 nodes, PF with $\mu = 0.6$ doubles the overlay size (282 nodes).

In summary, μ should be chosen large to protect the anonymity. The only exception applies to attribute overlays that are short living, i.e., are used to exchange one or very few notifications.

SUMMARY To summarize the research questions, the anonymity improves proportionally with the forwarding probability μ . The overlay size and the signaling overhead however also increases with μ . These metrics even double between $\mu = 0.4$ and $\mu = 0.8$.

5.3.6 Shell game parameter and attack accuracy

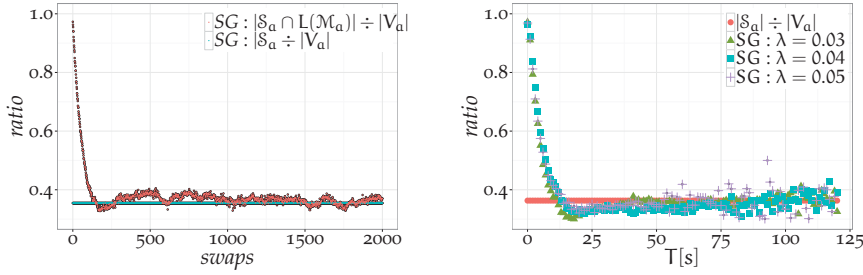
This section analyzes the SG with the following research questions:

1. How many SGs are necessary to hide leaf subscribers?
2. Which assignment of the exponential decay parameter λ for the SG ensures that leaf subscribers are hidden while maintaining a low signaling overhead?

The exponential decay function (cf. Section 4.8) controls the behavior of SG. Good values for the function parameter λ have to be discovered in order to protect subscriber anonymity against the attacker on one hand and to minimize the signaling overhead induced by the shell game on the other hand. The SG is simulated with varying λ , and the ratio of leaf subscribers among leaf nodes over time is measured as indicator for the protection of anonymity. Furthermore,

the global number of swaps is measured as indicator of the signaling overhead. The time indicates how quickly anonymity is achieved. The parameter is simulated within the interval $\lambda \in [0, 0.05]$. To prevent colliding swaps of adjacent nodes as well to reduce signaling costs, every node waits for a random time interval after each shell game. This time interval is drawn from the exponential distribution with mean 5s, i.e., the mean is greater than the inter heartbeat time of 1s.

To achieve a position mixture of forwarders and subscribers in the overlay, nodes are expected to be able to swap once across the diameter of the overlay. In case of a single publisher, this is the height of a tree. With basic overlay of 500 nodes and $|N(v)| = 4$, the overlay tree height is estimated to be within $\log_4(500) = 4.48$. In terms of total number of swaps, $141 \times 4.48 = 587$ (141 overlay nodes taken from experiment PF) swaps can be assumed to be sufficient to shuffle the whole overlay. As nodes play the SG independently, the duration until this state is reached can be estimated as $4.48 \times 5s = 22.4s$.



(a) SG. Leaf subscriber ratio $|S_a \cap L(\mathcal{M}_a)| / |V_a|$ in dependence on the global number of swaps compared to the optimal leaf subscriber ratio for privacy. Decay function in dependence on the local overlay membership time.

(b) SG. Leaf subscriber ratio in dependence on the simulation time, each for $\lambda = \{0.03, 0.04, 0.05\}$ compared to the optimal leaf subscriber ratio for privacy.

Figure 40: Convergence of the SG (1).

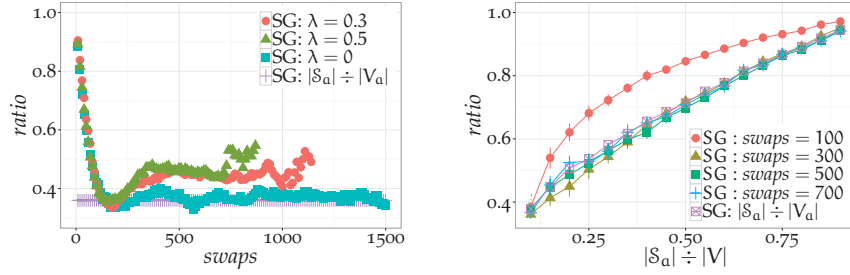
Figure 40a shows the average convergence of the leaf subscriber ratio towards the overlay subscriber ratio over the total number of swaps with $\lambda = 0$. The ratio converges indeed around 587 swaps. However, AnonPubSub with the SG seems to reach the optimal ratio before this convergence. This occurs as subscribers already play the shell game before the attribute mesh is connected. However, subscribers are likely to swap into leaf position as the diameter of overlay sections is low in this stage. Hence, the ratio of leaf subscriber fluctuates before reaching convergence.

Figure 40b shows the convergence for $\lambda \in \{0.03, 0.04, 0.05\}$ over time. The ratio of leaf subscribers ($|S_a \cap L(\mathcal{M}_a)| / |L(\mathcal{M}_a)|$) reaches the target ratio before the expected 22.4s, as for the number of swaps. However, the ratio of leaf subscribers converges slower to the optimal ratio than expected. This is due to collisions of swaps played by adjacent nodes, which causes delay for some swaps. Hence, it takes slightly longer

than 22.4s to swap 587 times (≈ 4.22 swaps initiated by every mesh node).

Higher values of λ , e.g. 0.05 (purple crosses in Figure 40b) scatter over time. This happens as the systems decay function lowers the number of swaps per second. The average ratio depends on fewer measurements. A value of $\lambda = 0.05$ starts to scatter once anonymity optimal accuracy is reached, $\lambda = 0.04$ scatters later, i.e., unnecessary swaps later. As a conclusion, $\lambda = 0.05$ seems to be a good value for low signaling overhead, whereas $\lambda = 0.04$ is a more conservative estimation.

Figure 41a illustrates why the decay function depends on each nodes attribute mesh membership time rather than the played swaps. By eliminating the influence of the decay function with $\lambda = 0$, SG works as expected (bottom plot). However, for a very high value of $\lambda = 0.5$ $|\mathcal{S}_a \cap L(\mathcal{M}_a)|/|L(\mathcal{M}_a)|$ it converges away from optimum (top plot). The same effect occurs for smaller values as used in Figure 41a. While the overlay is not connected, nodes close the leaf positions cool down early, thus leaving many subscribers back in leaf position.



(a) SG. Ratio of leaf subscribers in dependence on the global number of *swaps*, each for $\lambda = \{0, 0.3, 0.5\}$ compared to the optimal leaf subscriber ratio for privacy. Decay function in dependence on the local number of *swaps*. (b) SG. Ratio of leaf subscribers in dependence on the ratio of subscribers, each after *swaps* = {100, 300, 500, 700} compared to the optimal ratio of subscribers for privacy.

Figure 41: Convergence of the SG (2).

Figure 41b illustrates the high sensitivity of SG against the subscriber ratio (cf. Figure 37b). In particular, a massive increase in the number of swaps by lowering λ (compare 300 swaps with 700 swaps) cannot sufficiently compensate. The SG therefore has to be combined with probabilistic forwarding for higher subscriber ratios.

In summary, λ must be chosen very small to allow for sufficient node swaps. With the parameter selection at hand, a selection of $\lambda \leq 0.05$ works well. Higher heartbeat rates may allow for a higher λ . It is important to note that a high λ may cause nodes to swap back into their original position. This behaviour causes the SG to be counter-productive: message overhead occurs while no additional anonymity protection is achieved.

SUMMARY To summarize the research questions, every node needs to initiate about as many SGs as the height of a distribution tree (picking a single publisher as root) within the overlay—about 4 – 5 SGs. Still, nodes participate in more SGs, but not as initiator. For the basic overlay setup uses here, a decay with $\lambda = 0.04$ seems to be ideal to let every node initiate the SG about 5 times before becoming less active.

5.3.7 Probabilistic forwarding & shell game

While SG provides good protection against the global attacker for low ratios of subscribers, high ratios of overlay subscribers require probabilistic forwarding in addition. For instance, if the $|S|/|V|$ exceeds 0.25, 0.514 of all overlay nodes are subscribers (cf. Figure 41b). PF can be used to increase the ratio of overlay forwarders. The results in Figure 39b give an indicator how to adjust μ to lower the ratio of overlay subscribers below 0.5.

However, probabilistically selected neighbors have to act opportunistically as they may have no initial successors. Furthermore, as the total number of subscribers in the system is unknown to nodes in AnonPubSub due to anonymity protection, good parameterizations of the probability μ can only be estimated.

5.3.8 Request / response-based internal attacks

This section analyzes the influence of the malicious insiders capabilities with the introduced request/response-based attack. Certain parameters such as the topology of the basic overlay are expected to influence this attack. This section therefore addresses the following research questions:

1. What is the influence of the connection delay of the basic overlay on the attackers ability to determine the distance to the subscriber?
2. How much does the connectivity of malicious nodes \mathcal{C}_a , i.e., the number of neighbors, influence the information gain?
3. Does the distance between colluding nodes \mathcal{C}_a influence the information gain?
4. How much do additional controlled nodes, i.e., more collusion, improve the information gain?

To answer these questions, a setup is described hereafter that assumes that no prior attribute overlay \mathcal{M}_a exists as malicious nodes in \mathcal{C}_a may spoof messages to disassemble the overlay before each attack step. The attacker is a combination of global observer and (colluding) malicious insiders as introduced in Section 3.3, page 41. This attacker knows the topology of the basic overlay G as well as the subscriber ratio $|S_a|/|V|$. This attacker model is strong in comparison to related

PARAMETER	EXPLANATION
$ V = 200$	size of the basic overlay
$ S_a / V = 0.1$	ratio of subscribers
$ \mathcal{C}_a = 1$	number of colluding malicious nodes
$ \mathcal{C}_a / V = 0.001$	ratio adverse to genuine nodes
$ N(v)' \in [1, 5]$	neighbors used by attacker
$runs = 25$	repetitions per setup

Table 9: Default parameter settings for internal attacks.

anonymization services, which only use one of the stated capabilities. Given an attribute a the attacker tries to establish a set S'_a as an approximation of the real subscriber set S_a .

Specific settings and setting boundaries for these experiments are given by Table 9. All other settings remain as in Table 7. The following paragraph address the four research questions each.

END-TO-END DELAY To analyze the influence of connection delay between nodes on the correct estimation of the subscriber distance, the delay distribution as well as subscriber and attacker positions are varied. A single malicious insider determines the distance to the closest subscriber via each neighbor from $N_a(v)$. The settings are given by Table 9, with varying δ of uniform $\mathcal{U}(left, right)$ and normal $\mathcal{N}(mean, variance)$ distributions.

For this experiment, 200 repetitions per distribution with random placements each are used. Then the attacker's distance estimation $|path(v_0, v'_s)|$, with the $\mathcal{I}\mathcal{A}$ as v_0 and estimated subscriber v'_s , is compared with the real distance $|path(v_0, v_s)|$. The attacker uses the *Request Round-Trip* approach to approximate δ_{avg} .

The static delay serves as reference and a linear increase of the hop error rate for the normal distribution is expected as the average accumulated variance should not exceed one edge delay for several hops. Compared, the uniform distributions should quickly approach the error rate of a random guess.

Figure 42a shows the error rates in dependence of the distance $|path(v_0, v_s)|$ in between v_0 and v_s for the five delay distributions. As expected, the attacker well compensates normal distributed delay across the average basic overlay diameter of ≈ 6 hops. The uniformly distributed delay already produces high error rates from the second hop onwards.

Concluding, delays from a narrow normal distribution, e.g., within a LAN, have only minor impact on the correct hop-count estimation. However, delays from a uniform distribution effectively prevent the attack. A uniform delay with a parameter range that covers the diameter of the basic overlay renders the attack infeasible.

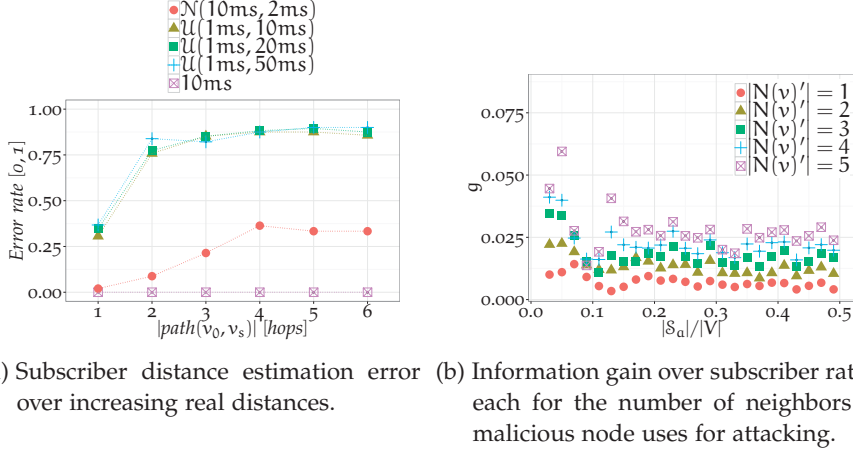


Figure 42: Influence of the basic overlay on the I2L.

ATTACKER CONNECTIVITY To analyze the influence of the attacker's node degree on the information gain, the number of neighbors the attacker uses is varied. The settings are as given in Table 9, with the ratio of subscribers $|S_a|/|V| \in (0, 0.5)$, and the attacker successively uses $|N(v_0)'| \in [1, 5]$ of his available $|N(v_0)|$ neighbors.

Increasing $|N(v)'|$ is expected to cause a linear increase in information gain as the attacker can exploit more attack paths in case no close cycles exist.

Figure 42b shows the information gain per subscriber over the subscriber ratio for the different number of used neighbors. For an increasing subscriber ratio, the attacker gains slightly less as more subscribers on the same path mask each other. However, a second neighbor almost doubles the information gain.

Concluding, the attacker's additional advantage of good connectivity becomes smaller with every neighbor. Furthermore, large subscriber ratios render the attack harder.

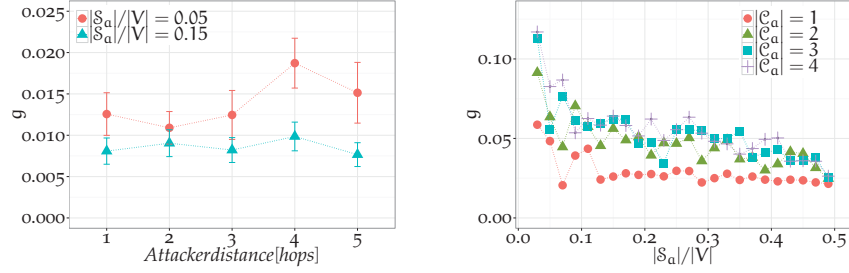
DISTANCE BETWEEN COLLUDING NODES Two malicious nodes $|C_a| = 2$ are used to evaluate the influence of the hop distance between attackers. Both malicious insiders are randomly placed and the results grouped by their distance. The information gain is measured as the combined result of both attackers.

The settings are as given in Table 9, simulated with a low and high subscriber ratio $|S_a|/|V| = \{0.05, 0.15\}$. The experiment is repeated 300 times to obtain sufficient samples for every attacker distance. Two random nodes are select v_x, v_y as adverse ones each. The results are grouped by the shortest path length $|path(v_x, v_y)|$ between the attackers. The information gain is used as metric.

First, a large subscriber ratio should yield to a lower information gain per subscriber as shown by the previous experiment. Second, a longer path between the attacking nodes should increase the information gain: given two attacking nodes v_x, v_y and a subscriber v_s , a longer path $|path(v_x, v_y)|$ should increase the probability of pairwise distinct nodes in $path(v_x, v_s)$ and $path(v_y, v_s)$. Hence, the attacker gets

more distance measurements to the subscriber and can rule out more forwarders.

Figure 43a shows the average information gain per subscriber over the distance between attacking nodes. An increasing distance between controlled nodes shows no significant improvement of the information gain. It appears that the attackers improve on distant subscribers, but loose on very close subscribers.



(a) Information gain over the distances of attacker v_x, v_y for $|S_a|/|V| = \{0.05, 0.15\}$. (b) Information gain of a colluding attacker over subscriber ratio. The additional gain drops with each additional node.

Figure 43: Anonymity attacks by the I2I with collusion.

Concluding, the distance between colluding nodes, and thus the placement of the attacker's nodes, has only minor impact on the attack outcome. Additional simulations conducted by us with higher graph diameters indicate a slight decrease of the gain for large distances.

COLLUSION OF MALICIOUS NODES Having analyzed the connectivity and distance of colluding attackers, the amount of colluding attackers is of relevance. For that, the number of colluding nodes is incremented successfully and each simulated with multiple subscriber ratios. The settings as given in Table 9 are used with subscriber ratios $|S_a|/|V| \in (0.0, 0.5)$. The number of malicious nodes is varied as $|\mathcal{C}_a| \in [1, 4]$.

More colluding attackers should obviously lead to higher information gain, similar to higher attacker connectivity: additional nodes should improve the information gain as the delay approximation becomes more accurate and as the attacker can identify more close by subscribers accurately. However, high numbers of attackers should also lead to more duplicate paths and thus the additional gain should be limited.

Figure 43b shows the information gain per subscriber of colluding nodes over the subscriber ratio. The attackers $|\mathcal{C}_a| = 2$ clearly improves over $|\mathcal{C}_a| = 1$, however the addition gain between $|\mathcal{C}_a| = 3$ and $|\mathcal{C}_a| = 4$ becomes smaller. Furthermore, the gain of additional nodes for very low ratios of subscribers is relatively low. The scenario with two attacking nodes causes larger confidence intervals due to the varying distance between both nodes over the 25 runs each (cf. Section 5.3.8).

Concluding, many colluding nodes only benefit scenarios with high subscriber ratios. Furthermore, the additional gain decreases with every additional node.

SUMMARY To summarize the request / response-based attack, the connectivity of an attacker seems to be the most relevant threat to anonymity. The number of colluding attackers appears to have a similar effect. That means, double the connectivity is equally beneficial for an attacker as double the number of controlled nodes. The distance between an attacker and a subscriber seems to have only a minor effect on the anonymity. The delay distribution of the basic overlay connections has a high impact on the anonymity. Nevertheless, even with a high delay variance, the anonymity attack is still effective.

5.3.9 Attacks based on churn and disconnecting nodes

This section analyzes the success of attacks against subscriber anonymity based upon churn and disconnecting nodes as published in [39]. Churn may cause forwarders to leave an attribute overlay, e.g., when their last subscriber left. Such behaviour can be used by the G \mathcal{A} to identify the forwarder as a forwarder and likewise reduce the anonymity sets of subscribers. An opportunistically acting forwarder may remain in the attribute overlay to protect the anonymity of other participants. The question of how much advantage an attacker obtains when forwarders are not acting opportunistically remains.

With disconnecting nodes, the I \mathcal{A} attempts to use the SG to obtain a position next to a leaf node. By interrupting the connection to the leaf node via the I \mathcal{A} , the G \mathcal{A} can distinguish non-opportunistic forwarders from subscribers, similar to churn. However, this attack is time consuming and may reveal the attacker. It is therefore important to know how beneficial this attack is in the first place.

This section answers the following research questions:

1. How do churn rates influence the attacker's information gain?
2. How much information gain can the attacker obtain with the node cornering attack, and what is the influence of PF?

IMPACT OF CHURN The effects of churn allow the global observer to distinguish subscribers from forwarders. However, it remains unclear how much the global observer gains from churn.

The settings from Table 7 are used for the simulation. The number of nodes $|V|$ as well as the ratio of nodes involved in churn $\varphi \in \{0.05, 0.1, 0.2\}$ is varied, and g is measured.

A linear correlation of g with φ is to be expected as the global observer can observe more overlay repairs with increasing φ . Furthermore, we g should increase in linear relation with $|V|$ due to longer path lengths in the overlay and thus more nodes affected by churn.

Figure 44a depicts the results of the simulation. As expected, g increases with φ for small amounts of nodes. However, the global

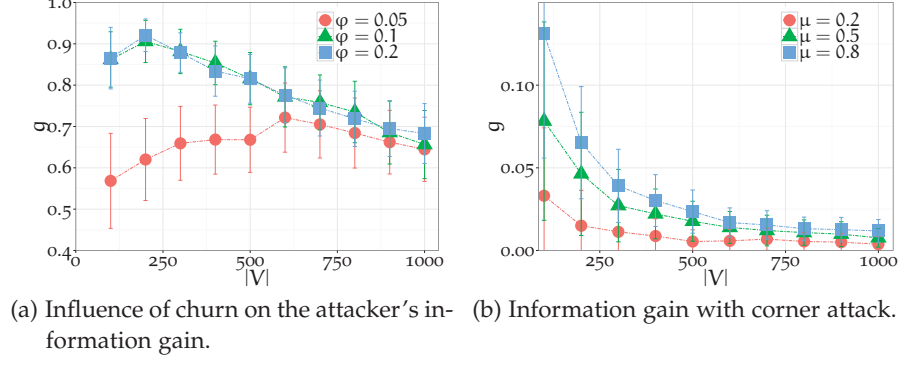


Figure 44: Anonymity attacks performed by I2I.

observer does not seem to benefit from high churn for higher number of nodes. This is because more subscribers become part of every $path_a(p, s)$. These nodes cause the path to remain stable (persists for many snapshots T). Hence, the global observer identifies less forwarders. In summary, the simulation results indicate that churn poses a high risk to nodes being de-anonymized.

NODE CORNERING With node cornering, the malicious insider exploits the position changes to explore the overlay guided by the global observer. The global observer observes the node behavior after the internal one disconnects them. However, it remains unclear how many cover nodes the attack can identify.

The same parameters as before are used, and the number of nodes $|V|$ as well as $\mu \in [0.2, 0.5, 0.8]$ is varied to measure g .

The gain g should increase with μ , as the attacker should be able to identify more cover nodes. Furthermore, a decrease in g with $|V|$ is expected due to a slower growth of $|path_a(p, s)|_{avg}$ compared to the growth of $|V|$.

Figure 44b depicts the results of the simulation. As expected, the gain increases with μ and drops quickly with increasing $|V|$. However, even with high μ , the overall gain for a single malicious insider remains low compared to other attacks.

In summary, this attack is hardly effective as a single malicious insider can only learn when moving into one branch of the overlay. The malicious insider can only compensate by waiting for other nodes to initiate position changes and thus “pull” the malicious insider back again. Alternatively, the malicious insider could collude. Still, the attack is invasive and could be detected by collaborating nodes.

5.3.10 Summary of the simulation

This section performed a quantitative evaluation of AnonPubSub by exposing the simulated system against the global observer G2I and the malicious insider I2I. For the latter one, a new timing-based attack were proposed that uses request / response semantics of an anonymous communication system. The attackers aim at breaking the

anonymity requirement, in particular subscriber anonymity. The evaluation of anonymity was complemented with an assessment of the signaling overhead that is caused by measures to protect anonymity.

The results showed that AnonPubSub without PF and the SG may protect against the malicious insider, but the global observer may still violate the anonymity. PF mitigates the success of the G \mathcal{A} partially. Increasing the parameter μ causes a linear decrease in the attacker's success (accuracy, information gain). This holds even for large subscriber ratios, i.e., when the advantages of PF diminish. However, a higher μ also causes an additional polynomial signaling overhead. Compared, the SG reduces the attacker's chances of de-anonymizing subscribers significantly. The best results are achieved with low ratios of subscribers as well as low node degree. The desired overlay randomization via SG is reached quickly. The decay function to reduce the signaling overhead of SG therefore works well. Moreover, SG and PF can be combined. This helps to compensate for the SG's lack with high ratios of subscribers.

Opposed to the global observer, the malicious insider can hardly de-anonymize subscribers, even without PF and SG. The results showed that for the malicious insider, high connectivity is as desirable as many colluding nodes. In particular, the additional gain of an additional malicious insider becomes quickly marginal. Randomized transmission delay between nodes impede the attacker significantly. However, even the addition of high random delay—magnitudes of the actual delay—does not fully prevent the attacker from reasoning about subscriber identities.

Concluding, the global attacker G \mathcal{A} poses the highest threat to anonymity. Given the system, the malicious insider I \mathcal{A} is highly dependent on the G \mathcal{A} to de-anonymize nodes. To mitigate anonymity attacks, opportunistic node behaviour is the key strategy. The two mechanisms PF and the SG mitigate anonymity attacks as well without relying on opportunistic node behaviour. However, the following crucial points have to be considered: PF works best with attribute overlays that have few forwarders, but causes significant messaging overhead with every notification. The SG works best with attribute overlays containing many forwarders, but requires an initial pause to garble the attribute overlay. A final takeaway point regarding anonymity attacks is the diameter of the attribute overlays: the presented attacks are most effective for low-diameter attribute overlays. However, with an increasing diameter, i.e., an increasing number of nodes, the success of such attacks reduces significantly.

5.4 PROOF-OF-CONCEPT: TWITTERIZE

This section introduces a proof of concept (POC) for the anonymous Pub/Sub system AnonPubSub as published in [35]. The POC implements the protocol introduced in the previous section as an Android app for micro-blogging as an instance of the citizen journalism scenario. This app integrates with Twitter but also protects confiden-

tiality and anonymity as well. For that, the app uses the Twitter social graph as basic membership, which can be used to send common tweets as well as anonymous tweets via the proposed protocol. This POC shows how AnonPubSub can be applied practically to a most prominent Pub/Sub application that demands for anonymity. The results indicate that only mild overhead is caused compared to non-anonymous communication.

Section 5.4.1 discusses related applications. Section 5.4.2 explains the building block selections and adaptations for the app. Section 5.4.3 concludes the discussion of the POC via an evaluation.

5.4.1 *Privacy-preserving twitter apps*

Twitter is a most popular example for citizen journalism with hundreds of million active users every month. Most significantly, Twitter played an important role on political movements such as the Arab Spring [93]. Here in particular anonymity of users is crucial to protect them from political persecution. As a result, several privacy-preserving Twitter alternatives have been proposed in recent years

Hummingbird [28] is an app that mimics Twitter functionality, but uses a separate infrastructure. Hummingbird encrypts tweets (confidentiality), provides access control, and allows users to follow hashtag via oblivious matching (subscription confidentiality). Anonymity is not protected by Hummingbird. Another app, Twister [58], introduces pseudonyms. Senders register a pseudonym, their nickname which is bound to a key pair via the Bitcoin [104] protocol. Pseudonyms enforce non-repudiation and cause all actions from the sender to be linkable. Senders publish signed tweets via the BitTorrent [26] protocol. In combination with an anonymization service, senders and receivers are unlinkable. The actions of a sender are however linkable. Twitsper [141, 142] uses Twitter in combination with a second central server to realize private group communication. Senders notify receivers via direct Twitter messages, the receivers then obtain further information from the Twitsper server. That means, Twitter can link senders and receivers, but cannot access messages. The second server may access messages, but cannot link senders and receivers. Senders and receivers must know each other for the key exchange. Besides confidential and anonymous communication, some approaches [140] perform metadata anonymization, i.e., prevent global observers from linking tweets and creating profiles. For that, techniques based upon k -anonymity [150] are used.

In summary, privacy-preserving Twitter as well as micro-blogging apps in general focus on confidentiality. Anonymity as defined in Section 3.2, page 39, can only be protected with add-on anonymization services.

BASIC MEMBERSHIP	TWITTER
V (nodes)	Twitter users
$N(v)^-$ (outgoing neighbors)	followers
$N(v)^+$ (incoming neighbors)	following (inverse $N(v)^-$)
$m_{\text{adv}t}$ (advertisement)	tweet
m_{sub} (subscription)	direct message
m_{notif} (notification)	tweet

Table 10: Basic membership management in Twitter

5.4.2 Twitterize

The Twitterize POC addresses several of the building blocks introduced in this thesis. First, the basic membership via Twitter is explained. Next, content distribution and attribute localization via AES and pseudonyms are summarized. Third, matching and key management via NFC and QR-code are introduced. Finally, an overview of the POC implementation for the Android platform is provided.

BASIC MEMBERSHIP MANAGEMENT The app uses the Twitter social graph as basic membership management. For that, Twitter users represent nodes V and followers represent the neighborhood set $N(v)^-$ with directed edges. Furthermore, tweets represent messages. Table 10 summarizes the mapping from the basic membership management notation to Twitter.

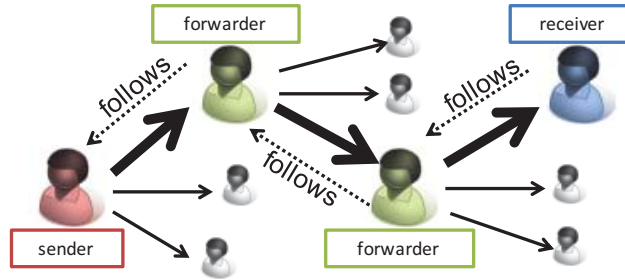


Figure 45: Twitter social graph with follow relation and tweet flow. The bold arrows represent the overlay for a hashtag on top of the social graph underlay.

Figure 45 depicts an example of a Twitter social graph. Users follow each other (dotted arrows) and thus receive status update tweets (solid arrows) from the users they follow. The app exploits this social graph as basic membership management and constructs hashtag overlays on top. In this example, the two forwarders in the center forward tweets from sender to receiver. Therefore, sender and receiver remain decoupled and no single forwarder can link sender and receiver together.

CONTENT DISTRIBUTION AND ATTRIBUTE LOCALIZATION Tweets are also used for content distribution. As each tweet is delivered to every follower—point to multi-point communication as opposed to point-to-point communication, it becomes harder for an attacker to determine the actual receiver for every communication hop. This behaviour of Twitter to some degree mimics the effect of anonymity protection mechanisms PF introduced in the previous chapter. Symmetric cryptography with AES in CBC mode protects the confidentiality.

Attributes are mapped to twitter hashtags. However, the app uses the encryption of the hash of each hashtag as pseudonym for attribute localization. Formally, the pseudonym t_a is defined as $t_a = \{H(a)\}_{K_a}$ where H is a hash function, and K_a a shared secret. Advertisements are flooded via tweets through the follower relation. As this relation is uni-directional, subscriptions are performed via direct messages.

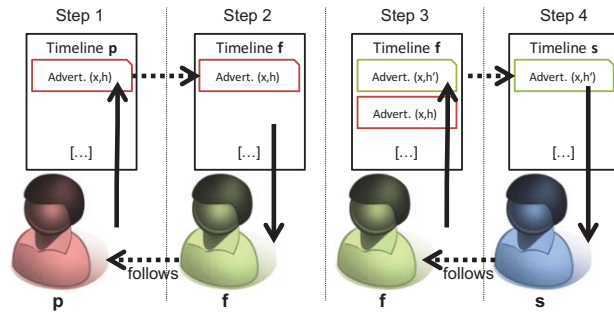


Figure 46: Advertisement tweets propagate in Twitter over timelines to followers. If an advertisement is new for a follower, the follower increments the hash element and posts it to her own timeline. Hence, the follower becomes a forwarder.

Figure 46 illustrates this process with Twitter. A publisher p tweets the advertisement as a status update to her timeline (step 1). As forwarder f follows p , the advertisement tweet will automatically show up on f 's timeline (step 2). Forwarder f forwards this advertisement by incrementing the hash element and posting another status update to her timeline (step 3). Finally, subscriber s receives the advertisement tweets as s follows f .

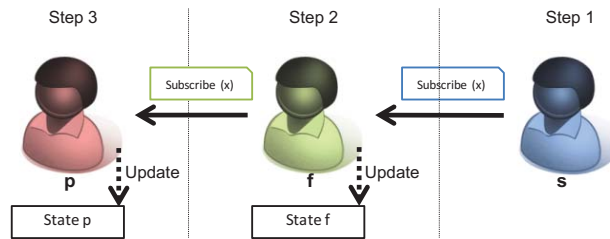


Figure 47: Subscription tweets propagate via direct tweets in Twitter. Users have to maintain a routing table as state outside of Twitter for the protocol.

Figure 47 illustrates this process from right to left. The subscriber s sends a subscription via a direct tweet to f (step 1). Forwarder f then updates its internal state, the routing table, and forwards the

subscription to p (step 2). Finally publisher p receives the subscription and updates her internal state as well. As direct tweets are being used here, the social graph may remain directed.

MATCHING AND KEY MANAGEMENT Matching is performed via the comparison of pseudonyms t_a . As the pseudonym is derived from a secret, publishers and subscribers are only required to share one secret. Subscribers can precompute the pseudonyms and thus are only required to perform a binary comparison.

The key management is performed out-of-band via [NFC](#) and [QR](#)-codes. Both mechanism are available to Android devices and allow the fast exchange of secrets via physical convergence of devices.

IMPLEMENTATION For the evaluation, the [POC](#) has been implemented for the Android mobile device platform as summarized in Figure 48. The implementation makes use of the Android [API](#), the [twitter4j](#) library², [AES](#) and an SQLite database³ as provided by the Android [APIs](#).

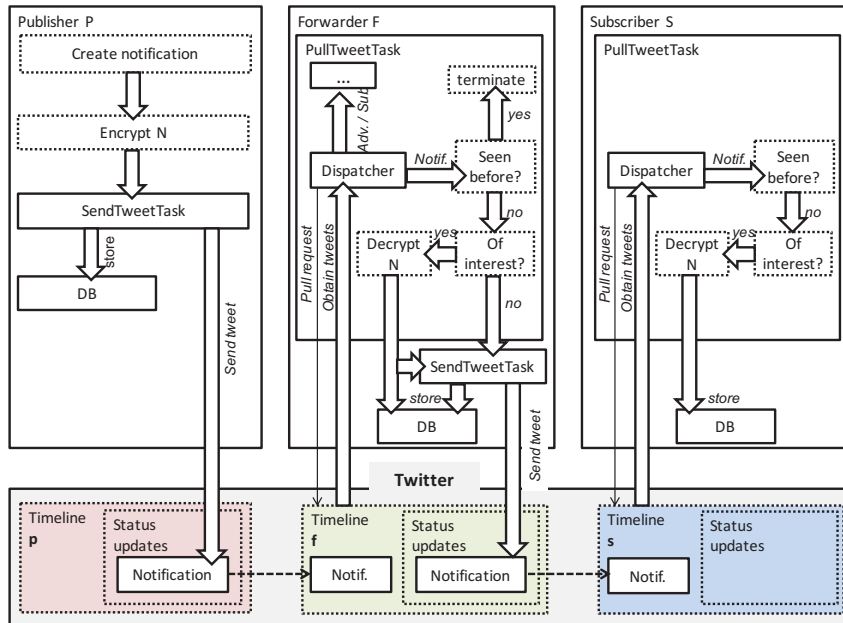


Figure 48: Software design of the Twitterize application. Parts of the design are shown as part of the information flow. From left to right: information flow and design for publishing tweets, forwarding tweets, and receiving tweets. The app is shown at the top, functions provided by Twitter at the bottom.

Distributing an anonymous and confidentiality notification works as follows: first, the publisher creates the notification, encrypts it with [AES](#) in cipher block chaining mode, attaches an attribute pseudonym, and sends the message as a tweet to Twitter. The publisher also notes down all this information in the database, and may also split the message in multiple parts to meet the length limit of a tweet as enforced by Twitter.

² <http://twitter4j.org>

³ <https://www.sqlite.org>

METRIC	TWITTERIZE	TWITTER
CPU load	8.454	7.739
Standard deviation (σ^2)	5.142%	3.906%
Power. consumption	2,216mW	2,117mW
Standard deviation (σ^2)	622mW	571mW

Table 11: Basic membership management in Twitter

Every application acts in the forwarder role and pulls new tweets on a regular basis from Twitter (*PullTweetTask*). It is then decided based on the tweet if this message is already known—discard—or new—forwarded. Known or duplicate tweets can occur as the Twitter social graph allows cycles and duplicate paths. To forward a message, it is again published as a tweet.

Every application with on or more subscriptions matches the attribute pseudonym in addition to forwarding. If the message matches, it is decrypted and stored in the database as well as displayed to the user.

5.4.3 Evaluation

This section evaluates the *POC* application with respect to the specific advantages and shortcomings of this implementation. First, empirical usage results are discussed in the context of an Android smartphone application. Second, the Twitter social graph is discussed as basic membership management.

EMPIRICAL USAGE To analyze the overhead that the app imposes over normal smartphone operations, CPU load and battery drain have been measured as means of processing overhead. For that, a LG Nexus 5 smartphone running Android OS version 4.4.3 as been used with the *Trepn Profiler* from Qualcomm⁴. The measurements have been compared with measurements for Twitter version 5.13.1 as reference application. The pull rate is set to $r = 1/60$ (once per minute), measured over 15 minutes in wake state with no user interaction, and obtained 8,850 samples each.

The results are summarized in Table 11. The mean measurements for Twitterize compared to Twitter are very close considering the high standard deviations. In summary, Twitterizes induces 9% higher CPU load and less than 5% additional power consumption.

The attribute localization process via flooding requires non-interested users to main some state information as forwarders and potential forwarders. However, the Android implementation requires to store 48Byte per hashtag. Hence, every participant can manage more than 21,000 hashtags per megabyte of storage space.

⁴ <https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler>

METRIC	LIMIT
Tweets	2,400 / day
Bulk tweets	15 / window
Window duration	15 minutes
Direct messages	250 / day
Timeline access	3,200

Table 12: Twitter limitations

The routing through a hashtag overlay causes delay of messages compared to normal Twitter usage. According to Watanabe and Suzumura [164], several samples of the Twitter social graph indicated a degree of separation of up to 4.71 users in 2012. Using this parameter for the Twitter underlay and pulling once per minute, the average delay is calculated as given in Equation (88) for a tweet with $d_{avg} = 142$ seconds ($d_{max} = 300$ seconds upper boundary) until delivery to all receivers.

$$d_{avg} = \frac{4.71}{2 \times r}; \quad r = \frac{1}{60} \quad (88)$$

TWITTER LIMITATIONS The selection of Twitter as basic membership management imposes some restrictions. Twitter limits the usage of their API to prevent abuse. Table 12 summarizes the limitations as of November 2014. The total number of tweets limits the number of notifications and advertisements. The bulk tweet limitation increases the delay during high usage times. The number of direct messages limits the number of subscription messages. The accessible timeline history restricts the number of messages the app can recover after connection loss / offline period. However, the tweet limit of 2,400 tweets per day restricts the app from forwarding messages more than the history of 3,200 tweets.

Figure 49 puts the tweet limitation into perspective of the number of attributes $|A|$ and notifications per attribute. Further insights regarding to overlay sizes can be found in Section 5.3.

5.4.4 Summary of the POC

This section has presented a real-world POC implementation of Anon-PubSub called Twitterize. Twitterize shows that AnonPubSub can be implement with various basic membership approaches, including an OSN such as Twitter. Moreover, the system works with tight resource constraints such as small message sizes and message per time limitations.

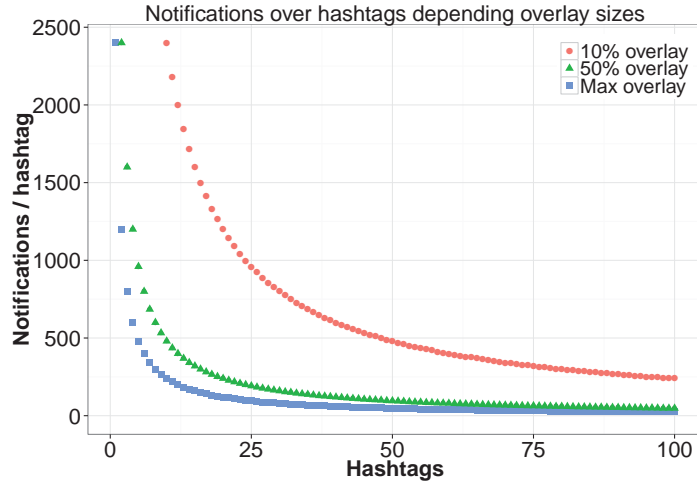


Figure 49: Tweets per hashtag over the number of hashtags in the system. The system degrades quickly with increasing overlay sizes, e.g., max overlay size. However, with an overlay membership rate of 10%, the system can still handle one tweet per hour for 1,000 hashtags.

5.5 SUMMARY

This chapter provided three means of evaluation for the anonymous [Pub/Sub](#) contributions introduced in the previous chapter. First, the qualitative discussion focussed on security and privacy requirements other than anonymity. Second, an extensive quantitative evaluation with a simulation exposed the contribution to anonymity attackers. Third, an empirical evaluation with a [POC](#) studied the scalability and overhead of the contribution.

The qualitative discussion showed that the presented contributions impede anonymity attacks performed by malicious insiders, similar to Darknets [24]. In particular, the absence of global node [IDs](#) prevents malicious insiders from gaining knowledge about nodes beyond their direct neighborhood. This aspect, which is also the basis for onion routing [44], makes anonymity attacks hard and requires ultimately the collusion with a global observing attacker. While the protection of confidentiality is not the focus of this thesis, the used mechanisms prevent attackers from learning the content of notifications. Furthermore, the privacy of subscriptions and advertisements is preserved against unauthorized participants. The presented system also ensures the scalability and protects the availability due to the mechanisms in place in the basic membership building block: as duplicate and looped messages may occur due to the anonymity of messages, effective measures are in place to detect and discard such messages quickly.

The simulation studies showed that the [SG](#) protects well against the anonymity attacker with global observing capabilities, even in collusion with the malicious insider. [PF](#) can be used to even the statistics of attribute overlays by increasing the size of anonymity sets. Several anonymity attacks were presented and evaluated as well. The pre-

sented anonymous Pub/Sub system seems to resist these attacks. Attacks performed by the global observer pose the highest threat to the anonymity of participants. While the malicious insider may augment these attacks, the process is slow and may be observed by benign participants. The key takeaway lessons of this study are:

- A short attribute overlay diameter benefits the anonymity attacker.
- High node churn also benefits the attacker. The benefit, however, reduces with the overlay diameter.
- PF demands a high forwarding probability $\mu \geq 0.5$ to effectively impede attacks. This causes significant messaging overhead.
- PF is most effective for attribute overlays with few nodes solely in the forwarder role.
- The SG effectively mitigates attacks assuming sufficient forwarders in the overlay.
- The SG requires some startup time to shuffle the overlay. This time, however, scales well with the number of nodes due to the parallel nature of the SG.
- Artificially introduced message delay has only limited effect on the attack mitigation.

The anonymous Pub/Sub system presented in this thesis is implemented as a mobile Twitter application for the citizen journalism scenario as well. This POC shows that the proposed mechanisms also work with a centralized and OSN-based network as basic membership and, therefore, supports the partitioning of anonymous Pub/Sub into building blocks. While this centralized basic overlay may be considered a SPoF for availability, it also avoids typical connectivity issues of mobile devices such as connection interruptions and network address translation (NAT). Moreover, the POC also uses a P2P key management well suited for mobile applications. The empirical study performed with this POC indicates that the signaling overhead required for anonymous Pub/Sub only increases the battery drain and CPU load mildly. A drawback arises from the tight usage restrictions enforced by Twitter: the scalability regarding the number of supported attributes and notifications remains limited.

CONCLUSION

This section recapitulates the work of this thesis. Section 6.1 summarizes the work and lists the key contributions and findings. Section 6.2 outlines future research directions that have been identified throughout the course of this thesis.

6.1 SUMMARY

This thesis tackled the challenge of fusing two research and technology areas, Pub/Sub and anonymous communication. Both areas are inherently hard to combine as they have diametrically conflicting properties: Pub/Sub depends on inspecting the messages in detail to route them to their destination. Anonymization explicitly prevents the inspection of message with cryptography and significantly limits, e.g., routing knowledge to the previous and next hop.

To meet this challenge, Chapter 2 first provided a foundation for both disciplines, Pub/Sub and anonymization services, as well as P2P systems. P2P systems seem to be a prime technology to construct systems with built-in anonymity. The main reason is that with P2P no single, or even central entity knows all participants of the system. Moreover, in particular with unstructured P2P, end-to-end (E2E) addressing of participants is not required; sending a message in a specific *direction* is sufficient. This thesis considered routing in a direction, therefore, as partitioning a P2P system into anonymity sets according to the number of possible directions. As another insight, Chapter 2 showed that splitting anonymization services from application has led to multiple attacks that rendered the anonymization service ineffective via application layer information. Anonymity and application on top are intrinsically related, and it is, therefore, desirable to construct both in conjunction.

Chapter 3 provided an overview of the related work. To structure this overview, seven building blocks were introduced to encapsulate the functions of anonymous Pub/Sub systems.

- Basic membership ensures connectivity between participants—the nodes in a network. The connectivity can be established, for instance, via gossiping algorithms. The encapsulation of basic membership into one building block enables the quick adaptation of anonymous Pub/Sub to different technologies. This is proven in the evaluation by the usage of two very different technologies: a general-purpose P2P system and a centralized system for mobile devices.
- Attribute localization connects publishers with subscribers, one of the most anonymous critical building blocks. It can be performed, for instance, via flooding. Two additional methods are

introduced in this thesis: a combination of the forest fire algorithm with a random walk, and a double random walk. This offering of attribute localization alternatives makes the presented anonymous Pub/Sub system extremely versatile as the alternative can be selected by the popularity of attributes to keep the message overhead as low as possible.

- Key management supplies publishers and subscribers with the keys necessary to encrypt, decrypt, sign, and verify signatures. Related work often assumes out-of-band key management. Key management, however, is also critical to anonymity. Two key management methods are adapted for anonymous Pub/Sub in this thesis: a centralized double-TTP and P2P key exchange via QR-codes and NFC. The first variation is well suited for Internet-connected devices. The second variation works best with personal trust and contact, for example, for citizen journalism.
- Matching takes care of matching subscriptions to notifications and advertisements. This function can become computationally expensive when combining combinatorial logic with matching. Furthermore, this function always leaks some information to anonymity attackers, whether messages match or not. However, while pseudonyms are used in this thesis for matching, the capsulation of matching within an independent building block allows the leveraging of vast amounts of research from the disciplines of cryptography and privacy-protection.
- Community management ensures that an established distribution network is connected and well functional, for instance, in the presence of node churn and node failures. Community management can be used to optimize attribute overlays, or the communities, given the appropriate mechanisms. Probabilistic forwarding (PF) and the shell game (SG) are introduced in this thesis as two mechanisms to optimize the goal of participant anonymity.
- Content distribution handles the distribution of notifications from publishers to subscribers and should be as efficient as possible, e.g., minimize the transmission delay. The way attribute overlays are established and optimized in this thesis make the content distribution extremely efficient. That is, notifications are not required to be delayed, cached, shuffled, and padded in any way. Moreover, the transmission rate of notifications is not restricted. These properties distinguish this contribution from many related approaches.

A novel anonymous and P2P Pub/Sub system was presented in Chapter 4. This system establishes overlay networks per attribute, an atomic concept to route message in Pub/Sub. Nodes are required to possess the appropriate attribute key material to participate as a publisher or subscriber in such an overlay. A key management, consisting of two TTPs, the anonymity TTP, and the attribute TTP, supplies

the nodes with the necessary keys. The overlay is designed in such a way that pure forwarders are part of the overlay and fulfill a similar role as proxy servers in anonymization services.

The constructed overlay networks protect against malicious insiders attempting to break anonymity. Compared to the related work, the this thesis focused on the construction of the overlay networks: Once the construction is complete, the additional overhead to protect anonymity becomes minimal. Two additional anonymity-enhancing technologies ensure that the system is also protected against a global observing attacker. Probabilistic forwarding (PF) pulls additional nodes into the overlay. Additional nodes increase the anonymity set sizes within the overlay on the one hand and prevent subscribers from being exposed in topologically easy to identify positions on the other hand. The shell game (SG), shuffles the overlay network by exchanging the positions of nodes. The SG ensures that an attacker cannot observe the position change per se, and can thus only reason about a randomized overlay. Both technologies in combination protect anonymity against global observing attackers. All contributions combined, anonymity is protected against the colluding combination of malicious insider and global observer. Related work typically only considers either one.

Overlay networks are subject to node churn over time, i.e., nodes joining and leaving. While overlays are constructed with fulfilling specific requirements in mind, e.g., distributing load evenly and low message transmission delay, these requirements can become invalid due to churn. To overcome this challenge, Chapter 4 also proposed a novel scheme for anonymous load balancing. The mechanism is based on random greedy group formation and circular exchange of counting Bloom filters. The greedy group formation works on local knowledge and does not require any prior information of load in the overlay. Counting Bloom filters serve as a probabilistic data structure to exchange load information while concealing the attributes for which a node is responsible. The same circular message exchange is then used by nodes to find matching partners to hand over the load, i.e., an attribute.

Chapter 5 complemented the mechanisms with an evaluation. For the evaluation, novel anonymity attacks were proposed. The request/-response attack can be used by a malicious insider to identify responders, e.g., subscribers, in a system. The attacker adaptively sends a request to different areas of the network and reasons about the responses. Candidates for the responses are stored in a probability distribution, incorporating how likely the response originated from a certain node. Repeated request/response attacks as well as collusion refine this distribution, ultimately exposing responders with high probability. The simulation results showed that the adaptive approach is highly successful. Furthermore, even a single attacker with high connectivity—many neighbors—can reach similar results as colluding attackers with low connectivity.

Anonymity enhancing technologies such as PF and the SG are effective in protecting the anonymity against malicious insiders as well

as against global observers. The protection of [PF](#), however, is limited and causes high signaling overhead. The [SG](#) protects anonymity very well within the limits of the node distributions in the overlay. Furthermore, the [SG](#) only causes mild signaling overhead. Combined with [PF](#), the node distributions within the overlay can be optimized while causing only a slight increase in signaling overhead.

The prototype Twitterize completed the evaluation with an Android application for the micro-blogging scenario. Twitterize establishes overlays per hashtag. Anonymous micro-blogging can be used in combination with normal micro-blogging. The prototype depends upon participants relaying messages in a [P2P](#) manner. For that, the application is required to run on the Android devices of all participants. To compensate for the potential loss of messages caused by unavailable participants, Twitterize uses the broadcast-like mechanisms of micro-blogging to send a message to all neighbors simultaneously. This reduces the possible loss or delay of messages. The results showed that computation and signaling overhead only cause mild battery drain compared to normal micro-blogging.

6.2 FUTURE WORK

This thesis covers the combination of anonymous communication and [Pub/Sub](#), with a particular focus on protecting subscriber anonymity. Several future research directions may be considered to complement the work of this thesis.

For one, the protection of sender anonymity, i.e., the anonymity of publishers, leads to manifold future challenges: How can publisher anonymity be protected without restricting message delay and rate? For instance, how can cover traffic be implemented and controlled to answer this question? Can malicious insiders abuse this system, and how can potential attacks be prevented?

As a second topic, the area of topological anonymity opens a complementary research area. Related work in the domain of [OSNs](#) has shown [143] that the local graph structure can reveal manifold information about [OSN](#) users. Likewise, the connectivity within the basic membership may reveal information.

A third topic is the key management. This thesis assumes closed groups, i.e., restricted key management. However, future application scenarios may require a less restricted key management via open groups. Open groups will give attackers new capabilities and thus require mechanisms for anonymity protection.

BIBLIOGRAPHY

- [1] Timothy G. Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price. „Browser-Based Attacks on Tor.“ In: *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*. Ed. by Nikita Borisov and Philippe Golle. Vol. 4776. Lecture Notes in Computer Science. Springer, 2007, pp. 184–199. DOI: [10.1007/978-3-540-75551-7_12](https://doi.org/10.1007/978-3-540-75551-7_12). URL: http://dx.doi.org/10.1007/978-3-540-75551-7_12.
- [2] A. Adams et al. *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*. RFC 3973 (Experimental). Internet Engineering Task Force, 2005. URL: <http://www.ietf.org/rfc/rfc3973.txt>.
- [3] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. „Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems.“ In: *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. IEEE Computer Society, 1979, pp. 218–223. DOI: [10.1109/SFCS.1979.34](https://doi.org/10.1109/SFCS.1979.34). URL: <http://dx.doi.org/10.1109/SFCS.1979.34>.
- [4] Ross J. Anderson. *Security engineering - a guide to building dependable distributed systems* (2. ed.) Wiley, 2008. ISBN: 978-0-470-06852-6.
- [5] Christer Andersson and Reine Lundin. „On the Fundamentals of Anonymity Metrics.“ In: *The Future of Identity in the Information Society - Proceedings of the Third IFIP WG 9.2, 9.6/ 11.6, 11.7/FIDIS International Summer School on The Future of Identity in the Information Society, Karlstad University, Sweden, August 4-10, 2007*. Ed. by Simone Fischer-Hübner, Penny Duquenoy, Albin Zuccato, and Leonardo A. Martucci. Vol. 262. IFIP Advances in Information and Communication Technology. Springer, 2007, pp. 325–341. DOI: [10.1007/978-0-387-79026-8_23](https://doi.org/10.1007/978-0-387-79026-8_23). URL: http://dx.doi.org/10.1007/978-0-387-79026-8_23.
- [6] Jean Bacon, David M. Eysers, Jatinder Singh, and Peter R. Pietzuch. „Access control in publish/subscribe systems.“ In: *Proceedings of the Second International Conference on Distributed Event-Based Systems, DEBS 2008, Rome, Italy, July 1-4, 2008*. Ed. by Roberto Baldoni. Vol. 332. ACM International Conference Proceeding Series. ACM, 2008, pp. 23–34. DOI: [10.1145/1385989.1385993](https://doi.org/10.1145/1385989.1385993). URL: <http://doi.acm.org/10.1145/1385989.1385993>.
- [7] Peter Bailey, Nick Craswell, and David Hawking. „Dark matter on the Web.“ In: *Poster Proceedings, 9th World-Wide Web Conference*. 2000, p. 2.

- [8] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. „Scalable application layer multicast.“ In: *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 19-23, 2002, Pittsburgh, PA, USA*. ACM, 2002, pp. 205–217. DOI: [10.1145/633025.633045](https://doi.org/10.1145/633025.633045). URL: <http://doi.acm.org/10.1145/633025.633045>.
- [9] Raphaël Barazzutti, Pascal Felber, Hugues Mercier, Emanuel Onica, and Etienne Riviere. „Thrifty privacy: efficient support for privacy-preserving publish/subscribe.“ In: *Proceedings of the Sixth ACM International Conference on Distributed Event-Based Systems, DEBS 2012, Berlin, Germany, July 16-20, 2012*. Ed. by François Bry, Adrian Paschke, Patrick Th. Eugster, Christof Fetzer, and Andreas Behrend. ACM, 2012, pp. 225–236. ISBN: 978-1-4503-1315-5. DOI: [10.1145/2335484.2335509](https://doi.org/10.1145/2335484.2335509). URL: <http://doi.acm.org/10.1145/2335484.2335509>.
- [10] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in database systems*. Vol. 370. Addison-wesley New York, 1987.
- [11] O. Berthold and H. Langos. „Dummy Traffic against Long Term Intersection Attacks.“ In: *PET*. Ed. by Roger Dingledine and Paul F. Syverson. Vol. 2482. Lecture Notes in Computer Science. Springer, 2002, pp. 110–128. ISBN: 3-540-00565-X.
- [12] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. „Web MIXes: A System for Anonymous and Unobservable Internet Access.“ In: *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*. Ed. by Hannes Federrath. Vol. 2009. Lecture Notes in Computer Science. Springer, 2000, pp. 115–129. DOI: [10.1007/3-540-44702-4_7](https://doi.org/10.1007/3-540-44702-4_7). URL: http://dx.doi.org/10.1007/3-540-44702-4_7.
- [13] Nabhendra Bisnik and Alhussein A. Abouzeid. „Modeling and analysis of random walk search algorithms in P2P networks.“ In: *Second International Workshop on Hot Topics in Peer-to-Peer Systems, HOT-P2P 2005, San Diego, California, USA, July 21, 2005*. IEEE Computer Society, 2005, pp. 95–103. ISBN: 0-7695-2417-6. DOI: [10.1109/HOT-P2P.2005.13](https://doi.org/10.1109/HOT-P2P.2005.13). URL: <http://dx.doi.org/10.1109/HOT-P2P.2005.13>.
- [14] Stevens Le Blond, Pere Manils, Chaabane Abdelberi, Mohamed Ali Kâafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. „One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users.“ In: *CoRR abs/1103.1518* (2011). URL: <http://arxiv.org/abs/1103.1518>.
- [15] Burton H. Bloom. „Space/Time Trade-offs in Hash Coding with Allowable Errors.“ In: *Commun. ACM* 13.7 (1970), pp. 422–426. DOI: [10.1145/362686.362692](https://doi.org/10.1145/362686.362692). URL: <http://doi.acm.org/10.1145/362686.362692>.

- [16] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. „Public Key Encryption with Keyword Search.“ In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 506–522. ISBN: 3-540-21935-8. DOI: [10.1007/978-3-540-24676-3_30](https://doi.org/10.1007/978-3-540-24676-3_30). URL: http://dx.doi.org/10.1007/978-3-540-24676-3_30.
- [17] Dan Boneh and Matthew K. Franklin. „Identity-Based Encryption from the Weil Pairing.“ In: *SIAM J. Comput.* 32.3 (2003), pp. 586–615. DOI: [10.1137/S0097539701398521](https://doi.org/10.1137/S0097539701398521). URL: <http://dx.doi.org/10.1137/S0097539701398521>.
- [18] Timothy A. Budd. *An introduction to object-oriented programming*. Addison-Wesley, 1991. ISBN: 978-0-201-54709-2.
- [19] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. „Design and evaluation of a wide-area event notification service.“ In: *ACM Trans. Comput. Syst.* 19.3 (2001), pp. 332–383. DOI: [10.1145/380749.380767](https://doi.org/10.1145/380749.380767). URL: <http://doi.acm.org/10.1145/380749.380767>.
- [20] Amina Chaabane, Codé Diop, Wassef Louati, Mohamed Jmaiel, Jorge R. Gomez-Montalvo, and Ernesto Exposito. „Towards a semantic-driven and scalable publish/subscribe framework.“ In: *IJIPT* 7.3 (2013), pp. 165–175. DOI: [10.1504/IJIPT.2013.055472](https://doi.org/10.1504/IJIPT.2013.055472). URL: <http://dx.doi.org/10.1504/IJIPT.2013.055472>.
- [21] David Chaum. „Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.“ In: *Commun. ACM* 24.2 (1981), pp. 84–88. DOI: [10.1145/358549.358563](https://doi.org/10.1145/358549.358563). URL: <http://doi.acm.org/10.1145/358549.358563>.
- [22] David Chaum. „The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability.“ In: *J. Cryptology* 1.1 (1988), pp. 65–75. DOI: [10.1007/BF00206326](https://doi.org/10.1007/BF00206326). URL: <http://dx.doi.org/10.1007/BF00206326>.
- [23] Sunoh Choi, Gabriel Ghinita, and Elisa Bertino. „A Privacy-Enhancing Content-Based Publish/Subscribe System Using Scalar Product Preserving Transformations.“ In: *Database and Expert Systems Applications, 21st International Conference, DEXA 2010, Bilbao, Spain, August 30 - September 3, 2010, Proceedings, Part I*. Ed. by Pablo Garcia Bringas, Abdelkader Hameurlain, and Gerald Quirchmayr. Vol. 6261. Lecture Notes in Computer Science. Springer, 2010, pp. 368–384. DOI: [10.1007/978-3-642-15364-8_32](https://doi.org/10.1007/978-3-642-15364-8_32). URL: http://dx.doi.org/10.1007/978-3-642-15364-8_32.
- [24] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. „Freenet: A Distributed Anonymous Information Storage and Retrieval System.“ In: *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity*

- and Unobservability, Berkeley, CA, USA, July 25-26, 2000, *Proceedings*. Ed. by Hannes Federrath. Vol. 2009. Lecture Notes in Computer Science. Springer, 2000, pp. 46–66. DOI: [10.1007/3-540-44702-4_4](https://doi.org/10.1007/3-540-44702-4_4). URL: http://dx.doi.org/10.1007/3-540-44702-4_4.
- [25] Sebastian Clauß and Stefan Schiffner. „Structuring anonymity metrics.“ In: *Proceedings of the 2006 Workshop on Digital Identity Management, Alexandria, VA, USA, November 3, 2006*. Ed. by Ari Juels, Marianne Winslett, and Atsuhiko Goto. ACM, 2006, pp. 55–62. DOI: [10.1145/1179529.1179539](https://doi.org/10.1145/1179529.1179539). URL: <http://doi.acm.org/10.1145/1179529.1179539>.
- [26] Bram Cohen. *The BitTorrent Protocol Specification*. Sept. 2014. URL: http://www.bittorrent.org/beps/bep_0003.html.
- [27] Giovanni Di Crescenzo, Brian A. Coan, John L. Schultz, Simon Tsang, and Rebecca N. Wright. „Privacy-Preserving Publish/Subscribe: Efficient Protocols in a Distributed Model.“ In: *Data Privacy Management and Autonomous Spontaneous Security - 8th International Workshop, DPM 2013, and 6th International Workshop, SETOP 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers*. Ed. by Joaquín García-Alfaro, Georgios V. Lioudakis, Nora Cuppens-Boulahia, Simon N. Foley, and William M. Fitzgerald. Vol. 8247. Lecture Notes in Computer Science. Springer, 2013, pp. 114–132. ISBN: 978-3-642-54567-2.
- [28] Emiliano De Cristofaro, Claudio Soriente, Gene Tsudik, and Andrew Williams. „Tweeting with Hummingbird: Privacy in Large-Scale Micro-Blogging OSNs.“ In: *IEEE Data Eng. Bull.* 35.4 (2012), pp. 93–100. URL: <http://sites.computer.org/debull/A12dec/humming.pdf>.
- [29] Common Criteria. *15408-2:2008 Information technology—Security techniques—Evaluation criteria for IT security—Part 2: Security functional components*. 2008. URL: <http://www.commoncriteriaportal.org/files/ccfiles/ccpart2v21.pdf>.
- [30] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David R. Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. „Building peer-to-peer systems with Chord, a distributed lookup service.“ In: *Proceedings of HotOS-VIII: 8th Workshop on Hot Topics in Operating Systems, May 20-23, 2001, Elmau/Oberbayern, Germany*. IEEE Computer Society, 2001, pp. 81–86. ISBN: 0-7695-1040-X. DOI: [10.1109/HOTOS.2001.990065](https://doi.org/10.1109/HOTOS.2001.990065). URL: <http://dx.doi.org/10.1109/HOTOS.2001.990065>.
- [31] George Danezis, Claudia Díaz, and Carmela Troncoso. „Two-Sided Statistical Disclosure Attack.“ In: *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*. Ed. by Nikita Borisov and Philippe Golle. Vol. 4776. Lecture Notes in Computer Science. Springer, 2007, pp. 30–44. DOI: [10.1007/978-3-540-75551-7_3](https://doi.org/10.1007/978-3-540-75551-7_3). URL: http://dx.doi.org/10.1007/978-3-540-75551-7_3.

- [32] George Danezis, Roger Dingledine, and Nick Mathewson. „Mixminion: Design of a Type III Anonymous Remailer Protocol.“ In: *2003 IEEE Symposium on Security and Privacy (S&P 2003)*, 11-14 May 2003, Berkeley, CA, USA. IEEE Computer Society, 2003, pp. 2–15. DOI: [10.1109/SECPRI.2003.1199323](https://doi.org/10.1109/SECPRI.2003.1199323). URL: <http://dx.doi.org/10.1109/SECPRI.2003.1199323>.
- [33] George Danezis and Andrei Serjantov. „Statistical Disclosure or Intersection Attacks on Anonymity Systems.“ In: *Information Hiding, 6th International Workshop, IH 2004, Toronto, Canada, May 23-25, 2004, Revised Selected Papers*. Ed. by Jessica J. Fridrich. Vol. 3200. Lecture Notes in Computer Science. Springer, 2004, pp. 293–308. DOI: [10.1007/978-3-540-30114-1_21](https://doi.org/10.1007/978-3-540-30114-1_21). URL: http://dx.doi.org/10.1007/978-3-540-30114-1_21.
- [34] Ajoy Kumar Datta, Maria Gradinariu, Michel Raynal, and Gwendal Simon. „Anonymous Publish/Subscribe in P2P Networks.“ In: *17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, 22-26 April 2003, Nice, France, CD-ROM/Abstracts Proceedings. IEEE Computer Society, 2003, p. 74. DOI: [10.1109/IPDPS.2003.1213174](https://doi.org/10.1109/IPDPS.2003.1213174). URL: <http://dx.doi.org/10.1109/IPDPS.2003.1213174>.
- [35] Jörg Daubert, Leon Böck, Panayotis Kikiras, Max Mühlhäuser, and Mathias Fischer. „Twitterize: Anonymous Micro-blogging.“ In: *11th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2014, Doha, Qatar, November 10-13, 2014*. IEEE, 2014, pp. 817–823. ISBN: 978-1-4799-7100-8. DOI: [10.1109/AICCSA.2014.7073285](https://doi.org/10.1109/AICCSA.2014.7073285). URL: <http://dx.doi.org/10.1109/AICCSA.2014.7073285>.
- [36] Jörg Daubert, Mathias Fischer, Tim Grube, Stefan Schiffner, Panayotis Kikiras, and Max Mühlhäuser. „AnonPubSub: Anonymous publish-subscribe overlays.“ In: *Elsevier Computer Communications* 76 (2016), pp. 42–53. ISSN: 0140-3664. DOI: [http://dx.doi.org/10.1016/j.comcom.2015.11.004](https://doi.org/10.1016/j.comcom.2015.11.004). URL: <http://www.sciencedirect.com/science/article/pii/S0140366415004211>.
- [37] Jörg Daubert, Mathias Fischer, Stefan Schiffner, and Max Mühlhäuser. „Distributed and Anonymous Publish-Subscribe.“ In: *Network and System Security - 7th International Conference, NSS 2013, Madrid, Spain, June 3-4, 2013. Proceedings*. Ed. by Javier Lopez, Xinyi Huang, and Ravi Sandhu. Vol. 7873. Lecture Notes in Computer Science. Springer, 2013, pp. 685–691. DOI: [10.1007/978-3-642-38631-2_57](https://doi.org/10.1007/978-3-642-38631-2_57). URL: http://dx.doi.org/10.1007/978-3-642-38631-2_57.
- [38] Jörg Daubert, Tim Grube, Max Mühlhäuser, and Mathias Fischer. „Internal attacks in anonymous publish-subscribe P2P overlays.“ In: *2015 International Conference and Workshops on Networked Systems, NetSys 2015, Cottbus, Germany, March 9-12, 2015*. IEEE, 2015, pp. 1–8. ISBN: 978-1-4799-5804-7. DOI: [10.1109/NetSys.2015.7134444](https://doi.org/10.1109/NetSys.2015.7134444).

- 1109/NetSys.2015.7089074. URL: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7083793>.
- [39] Jörg Daubert, Tim Grube, Max Mühlhäuser, and Mathias Fischer. „On the anonymity of privacy-preserving many-to-many communication in the presence of node churn and attacks.“ In: *13th Annual IEEE Consumer Communications and Networking Conference, CCNC 2016, Las Vegas, NV, USA, January 9-12, 2016*. IEEE, 2016, pp. 745–751. ISBN: 978-1-4673-9292-1.
- [40] Jörg Daubert, Alexander Wiesmaier, and Panayotis Kikiras. „A view on privacy & trust in IoT.“ In: *IEEE International Conference on Communication, ICC 2015, London, United Kingdom, June 8-12, 2015, Workshop Proceedings*. IEEE, 2015, pp. 2665–2670. DOI: [10.1109/ICCW.2015.7247581](https://doi.org/10.1109/ICCW.2015.7247581). URL: <http://dx.doi.org/10.1109/ICCW.2015.7247581>.
- [41] Nigel Deakin and Oracle America, Inc. *JSR-000914 Java Message Service (JMS) API*. 2002. URL: <https://jcp.org/aboutJava/communityprocess/final/jsr914>.
- [42] Stephen E. Deering and David R. Cheriton. „Multicast Routing in Datagram Internetworks and Extended LANs.“ In: *ACM Trans. Comput. Syst.* 8.2 (1990), pp. 85–110. DOI: [10.1145/78952.78953](https://doi.org/10.1145/78952.78953). URL: <http://doi.acm.org/10.1145/78952.78953>.
- [43] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. „Towards Measuring Anonymity.“ In: *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*. Ed. by Roger Dingledine and Paul F. Syverson. Vol. 2482. Lecture Notes in Computer Science. Springer, 2002, pp. 54–68. DOI: [10.1007/3-540-36467-6_5](https://doi.org/10.1007/3-540-36467-6_5). URL: http://dx.doi.org/10.1007/3-540-36467-6_5.
- [44] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. „Tor: The Second-Generation Onion Router.“ In: *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. Ed. by Matt Blaze. USENIX, 2004, pp. 303–320. URL: <http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html>.
- [45] Roger Dingledine, Vitaly Shmatikov, and Paul F. Syverson. „Synchronous Batching: From Cascades to Free Routes.“ In: *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers*. Ed. by David Martin and Andrei Serjantov. Vol. 3424. Lecture Notes in Computer Science. Springer, 2004, pp. 186–206. DOI: [10.1007/11423409_12](https://doi.org/10.1007/11423409_12). URL: http://dx.doi.org/10.1007/11423409_12.
- [46] Danny Dolev and Andrew Chi-Chih Yao. „On the security of public key protocols.“ In: *IEEE Transactions on Information Theory* 29.2 (1983), pp. 198–207. DOI: [10.1109/TIT.1983.1056650](https://doi.org/10.1109/TIT.1983.1056650). URL: <http://dx.doi.org/10.1109/TIT.1983.1056650>.

- [47] Technische Universität Dresden. *Project: AN.ON–Anonymity.Online*. Online, September 9th, 2015. 2015. URL: http://anon.inf.tu-dresden.de/index_en.html.
- [48] Matthew Edman and Bülent Yener. „On anonymity in an electronic society: A survey of anonymous communication systems.“ In: *ACM Comput. Surv.* 42.1 (2009). DOI: [10.1145/1592451.1592456](https://doi.org/10.1145/1592451.1592456). URL: <http://doi.acm.org/10.1145/1592451.1592456>.
- [49] Christian Esposito and Mario Ciampi. „On Security in Publish/Subscribe Services: A Survey.“ In: *IEEE Communications Surveys and Tutorials* 17.2 (2015), pp. 966–997. DOI: [10.1109/COMST.2014.2364616](https://doi.org/10.1109/COMST.2014.2364616). URL: <http://dx.doi.org/10.1109/COMST.2014.2364616>.
- [50] P. Th. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. „The many faces of publish/subscribe.“ In: *ACM Computing Surveys (CSUR)* 35.2 (June 2003), pp. 114–131. ISSN: 03600300. DOI: [10.1145/857076.857078](https://doi.org/10.1145/857076.857078).
- [51] Patrick Th. Eugster, Pascal Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. „The many faces of publish/subscribe.“ In: *ACM Comput. Surv.* 35.2 (2003), pp. 114–131. DOI: [10.1145/857076.857078](https://doi.org/10.1145/857076.857078). URL: <http://doi.acm.org/10.1145/857076.857078>.
- [52] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. „A Practical Congestion Attack on Tor Using Long Paths.“ In: *18th USENIX Security Symposium, Montreal, Canada, August 10–14, 2009, Proceedings*. Ed. by Fabian Monrose. USENIX Association, 2009, pp. 33–50. URL: http://www.usenix.org/events/sec09/tech/full_papers/evans.pdf.
- [53] Li Fan, Pei Cao, Jussara M. Almeida, and Andrei Z. Broder. „Summary cache: a scalable wide-area web cache sharing protocol.“ In: *IEEE/ACM Trans. Netw.* 8.3 (2000), pp. 281–293. DOI: [10.1109/90.851975](https://doi.org/10.1109/90.851975). URL: <http://dx.doi.org/10.1109/90.851975>.
- [54] John Fanning, Shawn Fanning, and Sean Parker. *napster*. Online, September 22nd, 2015. 1999. URL: <http://www.naptster.com>.
- [55] B. Fenner. *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*. RFC 4601 (Proposed Standard). Internet Engineering Task Force, 2006. URL: <http://www.ietf.org/rfc/rfc4601.txt>.
- [56] Amos Fiat and Moni Naor. „Broadcast Encryption.“ In: *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22–26, 1993, Proceedings*. Ed. by Douglas R. Stinson. Vol. 773. Lecture Notes in Computer Science. Springer, 1993, pp. 480–491. DOI: [10.1007/3-540-48329-2_40](https://doi.org/10.1007/3-540-48329-2_40). URL: http://dx.doi.org/10.1007/3-540-48329-2_40.

- [57] Michael J. Freedman and Robert Morris. „Tarzan: a peer-to-peer anonymizing network layer.“ In: *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*. Ed. by Vijayalakshmi Atluri. ACM, 2002, pp. 193–206. DOI: [10.1145/586110.586137](https://doi.org/10.1145/586110.586137). URL: <http://doi.acm.org/10.1145/586110.586137>.
- [58] Miguel Freitas. *twister Peer-to-peer microblogging*. Sept. 2014. URL: <http://twister.net.co>.
- [59] Sebastian Funke, Jörg Daubert, Alexander Wiesmaier, Panayotis Kikiras, and Max Mühlhäuser. „End-2-End Privacy Architecture for IoT.“ In: *IEEE Conference on Communications and Network Security, CNS 2015, poster proceedings, Florence, Italy, September 28-30, 2015*. IEEE, 2015, pp. 705–706. DOI: [10.1109/CNS.2015.7346895](https://doi.org/10.1109/CNS.2015.7346895).
- [60] Taher El Gamal. „A public key cryptosystem and a signature scheme based on discrete logarithms.“ In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472. DOI: [10.1109/TIT.1985.1057074](https://doi.org/10.1109/TIT.1985.1057074). URL: <http://dx.doi.org/10.1109/TIT.1985.1057074>.
- [61] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. „Peer-to-Peer Membership Management for Gossip-Based Protocols.“ In: *IEEE Trans. Computers* 52.2 (2003), pp. 139–149. DOI: [10.1109/TC.2003.1176982](https://doi.org/10.1109/TC.2003.1176982).
- [62] JonDos GmbH. *JonDonym Law enforcement*. Online, September 22nd, 2015. 2015. URL: https://anonymous-proxy-servers.net/en/law_enforcement.html.
- [63] Simon Godik, Anne Anderson, Bill Parducci, P Humenn, and S Vajjhala. *OASIS eXtensible access control 2 markup language (XACML) 3*. Tech. rep. Tech. rep., OASIS, 2002.
- [64] J.N. Gray. „Notes on data base operating systems.“ In: *Operating Systems*. Ed. by R. Bayer, R.M. Graham, and G. Seegmüller. Vol. 60. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1978, pp. 393–481. ISBN: 978-3-540-08755-7.
- [65] Ulrich Greveler, Benjamin Justus, and Dennis Loehr. „Forensic content detection through power consumption.“ In: *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*. IEEE, 2012, pp. 6759–6763. DOI: [10.1109/ICC.2012.6364822](https://doi.org/10.1109/ICC.2012.6364822). URL: <http://dx.doi.org/10.1109/ICC.2012.6364822>.
- [66] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. „King: estimating latency between arbitrary internet end hosts.“ In: *Computer Communication Review* 32.3 (2002), p. 11. DOI: [10.1145/571697.571700](https://doi.org/10.1145/571697.571700). URL: <http://doi.acm.org/10.1145/571697.571700>.

- [67] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. „SkipNet: A Scalable Overlay Network with Practical Locality Properties.“ In: *4th USENIX Symposium on Internet Technologies and Systems, USITS'03, Seattle, Washington, USA, March 26-28, 2003*. Ed. by Steven D. Gribble. USENIX, 2003. URL: <http://www.usenix.org/events/usits03/tech/harvey.html>.
- [68] Gaofeng He, Ming Yang, Xiaodan Gu, Junzhou Luo, and Yuanyuan Ma. „A novel active website fingerprinting attack against Tor anonymous system.“ In: *Proceedings of the IEEE 18th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2014, Hsinchu, Taiwan, May 21-23, 2014*. Ed. by Jiang-Liang Hou, Amy J. C. Trappey, Chien-Wei Wu, Kuo-Hao Chang, Chung-Shou Liao, Weiming Shen, Jean-Paul A. Barthès, and Jun-Zhou Luo. IEEE, 2014, pp. 112–117. DOI: [10.1109/CSCWD.2014.6846826](https://doi.org/10.1109/CSCWD.2014.6846826). URL: <http://dx.doi.org/10.1109/CSCWD.2014.6846826>.
- [69] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. „How much anonymity does network latency leak?“ In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM, 2007, pp. 82–91. ISBN: 978-1-59593-703-2. DOI: [10.1145/1315245.1315257](https://doi.org/10.1145/1315245.1315257). URL: <http://doi.acm.org/10.1145/1315245.1315257>.
- [70] Huiqi Hu, Yiqun Liu, Guoliang Li, Jianhua Feng, and Kian-Lee Tan. „A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions.“ In: *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. Ed. by Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman. IEEE, 2015, pp. 711–722. DOI: [10.1109/ICDE.2015.7113327](https://doi.org/10.1109/ICDE.2015.7113327). URL: <http://dx.doi.org/10.1109/ICDE.2015.7113327>.
- [71] "Information and Canada" Privacy Commissioner/Ontario. *Privacy by Design 7 Foundational Principles*. accessed January 22nd, 2015. 1995. URL: <http://www.privacybydesign.ca/index.php/about-pbd/7-foundational-principles/>.
- [72] Mihaela Ion, Giovanni Russello, and Bruno Crispo. „Supporting Publication and Subscription Confidentiality in Pub/Sub Networks.“ In: *Security and Privacy in Communication Networks - 6th International ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings*. Ed. by Sushil Jajodia and Jianying Zhou. Vol. 50. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2010, pp. 272–289. DOI: [10.1007/978-3-642-16161-2_16](https://doi.org/10.1007/978-3-642-16161-2_16). URL: http://dx.doi.org/10.1007/978-3-642-16161-2_16.

- [73] ISO/IEC. *Database language SQL (SQL-92)*, 9075:1992. 1992. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=16663.
- [74] ISO/IEC. *Common Criteria for Information Technology—Security Evaluation—Part2: Security functional requirements*. 1999. URL: <https://www.iso.org/obp/ui/%5C#iso:std:iso-iec:15408:-2:ed-3:v2:en>.
- [75] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley, 1991. ISBN: 978-0-471-50336-1.
- [76] Rob Jansen and Nicholas Hopper. „Shadow: Running Tor in a Box for Accurate and Efficient Experimentation.“ In: *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. URL: <http://www.internetsociety.org/shadow-running-tor-box-accurate-and-efficient-experimentation>.
- [77] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. „The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network.“ In: *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014. URL: <http://www.internetsociety.org/doc/sniper-attack-anonymously-deanonymizing-and-disabling-tor-network>.
- [78] K. R. Jayaram, Patrick Eugster, and Chamikara Jayalath. „Parametric Content-Based Publish/Subscribe.“ In: *ACM Trans. Comput. Syst.* 31.2 (2013), p. 4. DOI: [10.1145/2465346.2465347](https://doi.org/10.1145/2465346.2465347). URL: <http://doi.acm.org/10.1145/2465346.2465347>.
- [79] J. Jiangt and N. Skocik. „On the privacy protection in publish/-subscribe systems.“ In: *WCNIS. IEEE*, 2010, pp. 597–601. ISBN: 9781424458493.
- [80] Aaron Johnson, Paul F. Syverson, Roger Dingledine, and Nick Mathewson. „Trust-based anonymous communication: adversary models and routing algorithms.“ In: *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*. Ed. by Yan Chen, George Danezis, and Vitaly Shmatikov. ACM, 2011, pp. 175–186. DOI: [10.1145/2046707.2046729](https://doi.org/10.1145/2046707.2046729). URL: <http://doi.acm.org/10.1145/2046707.2046729>.
- [81] Dan Kaminsky. *Black Ops of TCP/IP 2011*. Tech. rep. White Ops, Inc., 2011. URL: <http://dankaminsky.com/2011/08/05/bo2k11/>.
- [82] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. „Limits of Anonymity in Open Environments.“ In: *Information Hiding, 5th International Workshop, IH 2002, Noordwijkerhout, The Netherlands, October 7-9, 2002, Revised Papers*. Ed. by Fabien A. P. Petitcolas. Vol. 2578. Lecture Notes in Computer Science. Springer,

- 2002, pp. 53–69. DOI: [10.1007/3-540-36415-3_4](https://doi.org/10.1007/3-540-36415-3_4). URL: http://dx.doi.org/10.1007/3-540-36415-3_4.
- [83] Dogan Kesdogan, Jan Egner, and Roland Büschkes. „Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System.“ In: *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*. Ed. by David Aucsmith. Vol. 1525. Lecture Notes in Computer Science. Springer, 1998, pp. 83–98. DOI: [10.1007/3-540-49380-8_7](https://doi.org/10.1007/3-540-49380-8_7). URL: http://dx.doi.org/10.1007/3-540-49380-8_7.
- [84] Mujtaba Khambatti, Kyung Dong Ryu, and Partha Dasgupta. „Structuring Peer-to-Peer Networks Using Interest-Based Communities.“ In: *Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003, Revised Papers*. Ed. by Karl Aberer, Vana Kalogeraki, and Manolis Koubarakis. Vol. 2944. Lecture Notes in Computer Science. Springer, 2003, pp. 48–63. ISBN: 3-540-20968-9. DOI: [10.1007/978-3-540-24629-9_5](https://doi.org/10.1007/978-3-540-24629-9_5). URL: http://dx.doi.org/10.1007/978-3-540-24629-9_5.
- [85] Joud Khoury, Gregory Lauer, Partha Pratim Pal, Bishal Thapa, and Joseph P. Loyall. „Efficient Private Publish-Subscribe Systems.“ In: *17th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2014, Reno, NV, USA, June 10-12, 2014*. IEEE, 2014, pp. 64–71. ISBN: 978-1-4799-4430-9. DOI: [10.1109/ISORC.2014.10](https://doi.org/10.1109/ISORC.2014.10). URL: <http://dx.doi.org/10.1109/ISORC.2014.10>.
- [86] Alexey Klimkin. *Unofficial eDonkey Protocol Specification vo.6.2*. 2004. URL: <http://www.jmule.org/files/eDonkey-protocol-0.6.2.html>.
- [87] Dejan Kostic, Adolfo Rodriguez, Jeannie R. Albrecht, Abhijeet Bhirud, and Amin Vahdat. „Using Random Subsets to Build Scalable Network Services.“ In: *4th USENIX Symposium on Internet Technologies and Systems, USITS'03, Seattle, Washington, USA, March 26-28, 2003*. Ed. by Steven D. Gribble. USENIX, 2003. URL: <http://www.usenix.org/events/usits03/tech/kostic.html>.
- [88] Jure Leskovec and Christos Faloutsos. „Sampling from large graphs.“ In: *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*. Ed. by Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos. ACM, 2006, pp. 631–636. DOI: [10.1145/1150402.1150479](https://doi.org/10.1145/1150402.1150479). URL: <http://doi.acm.org/10.1145/1150402.1150479>.
- [89] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. „Graphs over time: densification laws, shrinking diameters and possible explanations.“ In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*. Ed. by Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett. ACM,

- 2005, pp. 177–187. DOI: [10.1145/1081870.1081893](https://doi.org/10.1145/1081870.1081893). URL: <http://doi.acm.org/10.1145/1081870.1081893>.
- [90] Lawrence Lessig. *Code and other laws of cyberspace*. Basic books, 1999.
- [91] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. „t-Closeness: Privacy Beyond k-Anonymity and l-Diversity.“ In: *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*. Ed. by Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis. IEEE, 2007, pp. 106–115. DOI: [10.1109/ICDE.2007.367856](https://doi.org/10.1109/ICDE.2007.367856). URL: <http://dx.doi.org/10.1109/ICDE.2007.367856>.
- [92] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, and Weijia Jia. „A new cell counter based attack against tor.“ In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. Ed. by Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis. ACM, 2009, pp. 578–589. DOI: [10.1145/1653662.1653732](https://doi.org/10.1145/1653662.1653732). URL: <http://doi.acm.org/10.1145/1653662.1653732>.
- [93] Gilad Lotan, Erhardt Graeff, Mike Ananny, Devin Gaffney, Ian Pearce, et al. „The Arab Spring | the revolutions were tweeted: Information flows during the 2011 Tunisian and Egyptian revolutions.“ In: *International Journal of Communication* 5 (2011), p. 31.
- [94] OpenSim Ltd. *OMNeT++ simulation GUI*. Online, September 22nd, 2015. 2015. URL: <https://omnetpp.org/intro/screenshots>.
- [95] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. „A survey and comparison of peer-to-peer overlay network schemes.“ In: *IEEE Communications Surveys and Tutorials* 7.1-4 (2005), pp. 72–93. DOI: [10.1109/COMST.2005.1610546](https://doi.org/10.1109/COMST.2005.1610546). URL: <http://dx.doi.org/10.1109/COMST.2005.1610546>.
- [96] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. „Search and replication in unstructured peer-to-peer networks.“ In: *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2002, June 15-19, 2002, Marina Del Rey, California, USA*. ACM, 2002, pp. 258–259. DOI: [10.1145/511334.511369](https://doi.org/10.1145/511334.511369).
- [97] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. „Pseudonym Systems.“ In: *Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 9-10, 1999, Proceedings*. Ed. by Howard M. Heys and Carlisle M. Adams. Vol. 1758. Lecture Notes in Computer Science. Springer, 1999, pp. 184–199. ISBN: 3-540-67185-4. DOI: [10.1007/3-540-46513-8_14](https://doi.org/10.1007/3-540-46513-8_14). URL: http://dx.doi.org/10.1007/3-540-46513-8_14.

- [98] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. „L-diversity: Privacy beyond k -anonymity.“ In: *TKDD* 1.1 (2007). DOI: [10 . 1145 / 1217299 . 1217302](https://doi.org/10.1145/1217299.1217302). URL: [http : / / doi . acm . org / 10 . 1145 / 1217299 . 1217302](http://doi.acm.org/10.1145/1217299.1217302).
- [99] Abraham Harold Maslow. „A theory of human motivation.“ In: *Psychological review* 50.4 (1943), p. 370.
- [100] Darshan Mhapasekar. „Accomplishing anonymity in peer to peer network.“ In: *Proceedings of the 2011 International Conference on Communication, Computing & Security, ICCCS 2011, Odisha, India, February 12-14, 2011*. Ed. by Sanjay Kumar Jena, Rajeev Kumar, Ashok Kumar Turuk, and Manoranjan Dash. ACM, 2011, pp. 555–558. ISBN: 978-1-4503-0464-1. DOI: [10 . 1145 / 1947940 . 1948055](https://doi.org/10.1145/1947940.1948055). URL: <http://doi.acm.org/10.1145/1947940.1948055>.
- [101] Steven J. Murdoch and George Danezis. „Low-Cost Traffic Analysis of Tor.“ In: *2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May 2005, Oakland, CA, USA*. IEEE Computer Society, 2005, pp. 183–195. DOI: [10 . 1109 / SP . 2005 . 12](https://doi.org/10.1109/SP.2005.12). URL: <http://dx.doi.org/10.1109/SP.2005.12>.
- [102] M. Nabeel, N. Shang, and B. Elisa. „Efficient privacy preserving content based publish subscribe systems.“ In: *SACMAT*. ACM, 2012, pp. 133–144.
- [103] Mohamed Nabeel, Stefan Appel, Elisa Bertino, and Alejandro P. Buchmann. „Privacy Preserving Context Aware Publish Subscribe Systems.“ In: *Network and System Security - 7th International Conference, NSS 2013, Madrid, Spain, June 3-4, 2013. Proceedings*. Ed. by Javier Lopez, Xinyi Huang, and Ravi Sandhu. Vol. 7873. Lecture Notes in Computer Science. Springer, 2013, pp. 465–478. DOI: [10 . 1007 / 978 - 3 - 642 - 38631 - 2 _ 34](https://doi.org/10.1007/978-3-642-38631-2_34). URL: http://dx.doi.org/10.1007/978-3-642-38631-2_34.
- [104] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System (whitepaper)*. Sept. 2014. URL: <http://bitcoin.org/bitcoin.pdf>.
- [105] Arvind Narayanan and Vitaly Shmatikov. „Robust De-anonymization of Large Sparse Datasets.“ In: *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*. IEEE Computer Society, 2008, pp. 111–125. DOI: [10 . 1109 / SP . 2008 . 33](https://doi.org/10.1109/SP.2008.33). URL: <http://dx.doi.org/10.1109/SP.2008.33>.
- [106] Arvind Narayanan and Vitaly Shmatikov. „Myths and fallacies of "personally identifiable information".“ In: *Commun. ACM* 53.6 (2010), pp. 24–26.
- [107] Christiane Schulzki-Haddouti (Heise Online). *Nicht mehr ganz anonym: Anonymisier-Dienst JAP protokolliert Zugriffe*. Online, September 22nd, 2015 (German). 2003. URL: <http://www.heise.de/newsticker/meldung/Nicht-mehr-ganz-anonym-Anonymisier-Dienst-JAP-protokolliert-Zugriffe-83939.html>.

- [108] Christopher R. Palmer and J. Greg Steffan. „Generating network topologies that obey power laws.“ In: *Proceedings of the Global Telecommunications Conference, 2000. GLOBECOM 2000, San Francisco, CA, USA, 27 November - 1 December 2000*. IEEE, 2000, pp. 434–438. DOI: [10.1109/GLOCOM.2000.892042](https://doi.org/10.1109/GLOCOM.2000.892042). URL: <http://dx.doi.org/10.1109/GLOCOM.2000.892042>.
- [109] Jianli Pan and Raj Jain. *A survey of network simulation tools: Current status and future developments*. accessed August 21st, 2015. 2008. URL: <http://www.cse.wustl.edu/~jain/cse567-08/ftp/simtools/>.
- [110] Vasilis Pappas, Elias Athanasopoulos, Sotiris Ioannidis, and Evangelos P. Markatos. „Compromising Anonymity Using Packet Spinning.“ In: *Information Security, 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings*. Ed. by Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee. Vol. 5222. Lecture Notes in Computer Science. Springer, 2008, pp. 161–174. DOI: [10.1007/978-3-540-85886-7_11](https://doi.org/10.1007/978-3-540-85886-7_11). URL: http://dx.doi.org/10.1007/978-3-540-85886-7_11.
- [111] Sameer Parekh. „Prospects for remailers.“ In: *First Monday* 1.2 (1996).
- [112] Andreas Pfitzmann and Marit Hansen. *Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology*. 2005. URL: <http://freehaven.net/anonbib/cache/terminology.pdf>.
- [113] Andreas Pfitzmann and Marit Köhntopp. „Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology.“ In: *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*. Ed. by Hannes Federath. Vol. 2009. Lecture Notes in Computer Science. Springer, 2000, pp. 1–9. ISBN: 3-540-41724-9. DOI: [10.1007/3-540-44702-4_1](https://doi.org/10.1007/3-540-44702-4_1). URL: http://dx.doi.org/10.1007/3-540-44702-4_1.
- [114] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. „Accessing Nearby Copies of Replicated Objects in a Distributed Environment.“ In: *SPAA*. 1997, pp. 311–320. DOI: [10.1145/258492.258523](https://doi.org/10.1145/258492.258523). URL: <http://doi.acm.org/10.1145/258492.258523>.
- [115] György Pongor. „OMNeT: Objective Modular Network Testbed.“ In: *MASCOTS '93, Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems, January 17-20, 1993, La Jolla, San Diego, CA, USA*. Ed. by Herbert D. Schwetman, Jean C. Walrand, Kallol Kumar Bagchi, and Doug DeGroot. The Society for Computer Simulation, 1993, pp. 323–326.
- [116] Ralph Spencer Poore. „Anonymity, Privacy, and Trust.“ In: *Information Systems Security* 8.3 (1999), pp. 16–20.

- [117] Sandro Rafaeli and David Hutchison. „A survey of key management for secure group communication.“ In: *ACM Comput. Surv.* 35.3 (2003), pp. 309–329. DOI: [10.1145/937503.937506](https://doi.org/10.1145/937503.937506). URL: <http://doi.acm.org/10.1145/937503.937506>.
- [118] Costin Raiciu and David S. Rosenblum. „Enabling Confidentiality in Content-Based Publish/Subscribe Infrastructures.“ In: *Second International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm 2006, Baltimore, MD, Aug. 28 2006 - September 1, 2006*. IEEE, 2006, pp. 1–11. ISBN: 1-4244-0423-1. DOI: [10.1109/SECCOMW.2006.359552](https://doi.org/10.1109/SECCOMW.2006.359552). URL: <http://doi.ieeecomputersociety.org/10.1109/SECCOMW.2006.359552>.
- [119] Jean-François Raymond. „Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems.“ In: *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*. Ed. by Hannes Federrath. Vol. 2009. Lecture Notes in Computer Science. Springer, 2000, pp. 10–29. DOI: [10.1007/3-540-44702-4_2](https://doi.org/10.1007/3-540-44702-4_2). URL: http://dx.doi.org/10.1007/3-540-44702-4_2.
- [120] Michael K. Reiter and Aviel D. Rubin. „Crowds: Anonymity for Web Transactions.“ In: *ACM Trans. Inf. Syst. Secur.* 1.1 (1998), pp. 66–92. DOI: [10.1145/290163.290168](https://doi.org/10.1145/290163.290168). URL: <http://doi.acm.org/10.1145/290163.290168>.
- [121] Marc Rennhard and Bernhard Plattner. „Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection.“ In: *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society, WPES 2002, Washington, DC, USA, November 21, 2002*. Ed. by Sushil Jajodia and Pierangela Samarati. ACM, 2002, pp. 91–102. DOI: [10.1145/644527.644537](https://doi.org/10.1145/644527.644537). URL: <http://doi.acm.org/10.1145/644527.644537>.
- [122] Alfréd Rényi. „On the foundations of information theory.“ In: *Revue de l'Institut International de Statistique* (1965), pp. 1–14.
- [123] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.“ In: *Commun. ACM* 21.2 (1978), pp. 120–126. DOI: [10.1145/359340.359342](https://doi.org/10.1145/359340.359342). URL: <http://doi.acm.org/10.1145/359340.359342>.
- [124] Luis Roalter, Matthias Kranz, and Andreas Möller. „A Middleware for Intelligent Environments and the Internet of Things.“ In: *Ubiquitous Intelligence and Computing - 7th International Conference, UIC 2010, Xi'an, China, October 26-29, 2010. Proceedings*. Ed. by Zhiwen Yu, Ramiro Liscano, Guanling Chen, Daqing Zhang, and Xingshe Zhou. Vol. 6406. Lecture Notes in Computer Science. Springer, 2010, pp. 267–281. DOI: [10.1007/978-3-642-16355-5_23](https://doi.org/10.1007/978-3-642-16355-5_23). URL: http://dx.doi.org/10.1007/978-3-642-16355-5_23.

- [125] Antony I. T. Rowstron and Peter Druschel. „Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.“ In: *Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12-16, 2001, Proceedings*. Ed. by Rachid Guerraoui. Vol. 2218. Lecture Notes in Computer Science. Springer, 2001, pp. 329–350. ISBN: 3-540-42800-3. DOI: [10.1007/3-540-45518-3_18](https://doi.org/10.1007/3-540-45518-3_18). URL: http://dx.doi.org/10.1007/3-540-45518-3_18.
- [126] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. „SCRIBE: The Design of a Large-Scale Event Notification Infrastructure.“ In: *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*. Ed. by Jon Crowcroft and Markus Hofmann. Vol. 2233. Lecture Notes in Computer Science. Springer, 2001, pp. 30–43. DOI: [10.1007/3-540-45546-9_3](https://doi.org/10.1007/3-540-45546-9_3). URL: http://dx.doi.org/10.1007/3-540-45546-9_3.
- [127] Comuto SA. *Bla Bla Car*. Online, September 9th, 2015. 2015. URL: <https://www.blablacar.de>.
- [128] Amit Sahai and Brent Waters. „Fuzzy Identity-Based Encryption.“ In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Springer, 2005, pp. 457–473. DOI: [10.1007/11426639_27](https://doi.org/10.1007/11426639_27). URL: http://dx.doi.org/10.1007/11426639_27.
- [129] Stefan Schiffner and Sebastian Clauß. „Using Linkability Information to Attack Mix-Based Anonymity Services.“ In: *Privacy Enhancing Technologies, 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009, Proceedings*. Ed. by Ian Goldberg and Mikhail J. Atallah. Vol. 5672. Lecture Notes in Computer Science. Springer, 2009, pp. 94–107. ISBN: 978-3-642-03167-0.
- [130] Brian Krebs (Krebs on Security). *Online Cheating Site Ashley-Madison Hacked*. Online, July 15th, 2015. 2015. URL: <http://krebsonsecurity.com/2015/07/online-cheating-site-ashleymadison-hacked/>.
- [131] Marius Senftleben, Mihai Bucicoiu, Erik Tews, Frederik Armknecht, Stefan Katzenbeisser, and Ahmad-Reza Sadeghi. „MoP-2-MoP - Mobile Private Microblogging.“ In: *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Vol. 8437. Lecture Notes in Computer Science. Springer, 2014, pp. 384–396. DOI: [10.1007/978-3-662-45472-5_25](https://doi.org/10.1007/978-3-662-45472-5_25). URL: http://dx.doi.org/10.1007/978-3-662-45472-5_25.

- [132] Andrei Serjantov and George Danezis. „Towards an Information Theoretic Metric for Anonymity.“ In: *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*. Ed. by Roger Dingledine and Paul F. Syverson. Vol. 2482. Lecture Notes in Computer Science. Springer, 2002, pp. 41–53. DOI: [10.1007/3-540-36467-6_4](https://doi.org/10.1007/3-540-36467-6_4). URL: http://dx.doi.org/10.1007/3-540-36467-6_4.
- [133] C.E. Shannon. „A mathematical theory of communication.“ In: *Bell System Technical Journal, The* 27.3 (July 1948), pp. 379–423. ISSN: 0005-8580. DOI: [10.1002/j.1538-7305.1948.tb01338.x](https://doi.org/10.1002/j.1538-7305.1948.tb01338.x).
- [134] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. „P5: A Protocol for Scalable Anonymous Communication.“ In: *2002 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 12-15, 2002*. IEEE Computer Society, 2002, pp. 58–70. DOI: [10.1109/SECPRI.2002.1004362](https://doi.org/10.1109/SECPRI.2002.1004362). URL: <http://dx.doi.org/10.1109/SECPRI.2002.1004362>.
- [135] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. „P5: A protocol for scalable anonymous communication.“ In: *Journal of Computer Security* 13.6 (2005), pp. 839–876. URL: <http://content.iospress.com/articles/journal-of-computer-security/jcs245>.
- [136] Abdullatif Shikfa, Melek Önen, and Refik Molva. „Privacy in context-based and epidemic forwarding.“ In: *10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WOWMOM 2009, Kos Island, Greece, 15-19 June, 2009*. IEEE, 2009, pp. 1–7. DOI: [10.1109/WOWMOM.2009.5282445](https://doi.org/10.1109/WOWMOM.2009.5282445). URL: <http://dx.doi.org/10.1109/WOWMOM.2009.5282445>.
- [137] Abdullatif Shikfa, Melek Önen, and Refik Molva. „Privacy-Preserving Content-Based Publish/Subscribe Networks.“ In: *Emerging Challenges for Security, Privacy and Trust, 24th IFIP TC 11 International Information Security Conference, SEC 2009, Pafos, Cyprus, May 18-20, 2009. Proceedings*. Ed. by Dimitris Gritzalis and Javier Lopez. Vol. 297. IFIP Advances in Information and Communication Technology. Springer, 2009, pp. 270–282. ISBN: 978-3-642-01243-3. DOI: [10.1007/978-3-642-01244-0_24](https://doi.org/10.1007/978-3-642-01244-0_24). URL: http://dx.doi.org/10.1007/978-3-642-01244-0_24.
- [138] Abdullatif Shikfa, Melek Önen, and Refik Molva. „Privacy and confidentiality in context-based and epidemic forwarding.“ In: *Computer Communications* 33.13 (2010), pp. 1493–1504. DOI: [10.1016/j.comcom.2010.04.035](https://doi.org/10.1016/j.comcom.2010.04.035). URL: <http://dx.doi.org/10.1016/j.comcom.2010.04.035>.
- [139] Abdullatif Shikfa, Melek Önen, and Refik Molva. „Broker-Based Private Matching.“ In: *Privacy Enhancing Technologies - 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*. Ed. by Simone Fischer-Hübner and Nicholas Hopper. Vol. 6794. Lecture Notes in Computer Science. Springer, 2011, pp. 264–284. ISBN: 978-3-642-22262-7. DOI:

- 10.1007/978-3-642-22263-4_15. URL: http://dx.doi.org/10.1007/978-3-642-22263-4_15.
- [140] Amardeep Singh, Divya Bansal, and Sanjeev Sofat. „An Approach of Privacy Preserving based Publishing in Twitter.“ In: *Proceedings of the 7th International Conference on Security of Information and Networks, Glasgow, Scotland, UK, September 9-11, 2014*. Ed. by Ron Poet and Muttukrishnan Rajarajan. ACM, 2014, p. 39. ISBN: 978-1-4503-3033-6. DOI: 10.1145/2659651.2659733. URL: <http://doi.acm.org/10.1145/2659651.2659733>.
- [141] Indrajeet Singh, Michael Butkiewicz, Harsha V. Madhyastha, Srikanth V. Krishnamurthy, and Sateesh Addepalli. „Enabling private conversations on Twitter.“ In: *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*. Ed. by Robert H’obbes’ Zakon. ACM, 2012, pp. 409–418. ISBN: 978-1-4503-1312-4. DOI: 10.1145/2420950.2421009. URL: <http://doi.acm.org/10.1145/2420950.2421009>.
- [142] Indrajeet Singh, Michael Butkiewicz, Harsha V. Madhyastha, Srikanth V. Krishnamurthy, and Sateesh Addepalli. „Twitsper: Tweeting Privately.“ In: *IEEE Security & Privacy* 11.3 (2013), pp. 46–50. DOI: 10.1109/MSP.2013.3. URL: <http://dx.doi.org/10.1109/MSP.2013.3>.
- [143] Lisa Singh and Justin Zhan. „Measuring Topological Anonymity in Social Networks.“ In: *GrC. IEEE*, 2007, pp. 770–774.
- [144] Slashdot. *Ashley Madison Hack Claims First Victims*. Online, September 9th, 2015. 2015. URL: <http://yro.slashdot.org/story/15/08/24/1537259/ashley-madison-hack-claims-first-victims>.
- [145] Robin Snader and Nikita Borisov. „A Tune-up for Tor: Improving Security and Performance in the Tor Network.“ In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*. The Internet Society, 2008. URL: http://www.isoc.org/isoc/conferences/ndss/08/papers/03_tune-up_for_TOR.pdf.
- [146] Mudhakar Srivatsa and Ling Liu. „Securing publish-subscribe overlay services with EventGuard.“ In: *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*. Ed. by Vijay Atluri, Catherine Meadows, and Ari Juels. ACM, 2005, pp. 289–298. ISBN: 1-59593-226-7. DOI: 10.1145/1102120.1102158. URL: <http://doi.acm.org/10.1145/1102120.1102158>.
- [147] Chandler Harrison Stevens. „Many-to many communication.“ In: (1981).
- [148] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. „Chord: A scalable peer-to-peer lookup service for internet applications.“ In: *SIGCOMM*. 2001, pp. 149–

160. DOI: [10.1145/383059.383071](https://doi.org/10.1145/383059.383071). URL: <http://doi.acm.org/10.1145/383059.383071>.
- [149] Daniel Stutzbach and Reza Rejaie. „Understanding churn in peer-to-peer networks.“ In: *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeiro, Brazil, October 25-27, 2006*. Ed. by Jussara M. Almeida, Virgílio A. F. Almeida, and Paul Barford. ACM, 2006, pp. 189–202. DOI: [10.1145/1177080.1177105](https://doi.org/10.1145/1177080.1177105). URL: <http://doi.acm.org/10.1145/1177080.1177105>.
- [150] Latanya Sweeney. „k-Anonymity: A Model for Protecting Privacy.“ In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 557–570. DOI: [10.1142/S0218488502001648](https://doi.org/10.1142/S0218488502001648). URL: <http://dx.doi.org/10.1142/S0218488502001648>.
- [151] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. „Anonymous Connections and Onion Routing.“ In: *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*. IEEE Computer Society, 1997, pp. 44–54. DOI: [10.1109/SECPRI.1997.601314](https://doi.org/10.1109/SECPRI.1997.601314). URL: <http://dx.doi.org/10.1109/SECPRI.1997.601314>.
- [152] Andrew S. Tanenbaum and Maarten van Steen. *Distributed systems - principles and paradigms* (2. ed.) Pearson Education, 2007. ISBN: 978-0-13-239227-3.
- [153] Muhammad Adnan Tariq, Boris Koldehofe, Ala’ Altaweel, and Kurt Rothermel. „Providing basic security mechanisms in broker-less publish/subscribe systems.“ In: *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, Cambridge, United Kingdom, July 12-15, 2010*. Ed. by Jean Bacon, Peter R. Pietzuch, Joe Sventek, and Ugur Çetintemel. ACM, 2010, pp. 38–49. ISBN: 978-1-60558-927-5. DOI: [10.1145/1827418.1827425](https://doi.org/10.1145/1827418.1827425). URL: <http://doi.acm.org/10.1145/1827418.1827425>.
- [154] Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel. „Securing Broker-Less Publish/Subscribe Systems Using Identity-Based Encryption.“ In: *IEEE Trans. Parallel Distrib. Syst.* 25.2 (2014), pp. 518–528. DOI: [10.1109/TPDS.2013.256](https://doi.org/10.1109/TPDS.2013.256). URL: <http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.256>.
- [155] The Guardian (Sam Thielman). *Ashley Madison CEO Noel Biderman resigns after third leak of emails*. Online, August 28th, 2015. 2015. URL: <http://www.theguardian.com/technology/2015/aug/28/ashley-madison-neil-biderman-stepping-down>.
- [156] The New York Times. *Ashley Madison Users Face Threats if Blackmail and Identity Theft*. Online, September 9th, 2015. 2015. URL: http://www.nytimes.com/2015/08/28/technology/ashley-madison-users-face-threats-of-blackmail-and-identity-theft.html?_r=0.

- [157] Raphael Manfredi Tor Klingberg. *Gnutella 0.6*. Online, October 2nd, 2015. 2015. URL: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.
- [158] Carmela Troncoso, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. „Perfect Matching Disclosure Attacks.“ In: *Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*. Ed. by Nikita Borisov and Ian Goldberg. Vol. 5134. Lecture Notes in Computer Science. Springer, 2008, pp. 2–23. ISBN: 978-3-540-70629-8. DOI: [10.1007/978-3-540-70630-4_2](https://doi.org/10.1007/978-3-540-70630-4_2). URL: http://dx.doi.org/10.1007/978-3-540-70630-4_2.
- [159] András Varga et al. „The OMNeT++ discrete event simulation system.“ In: *Proceedings of the European simulation multiconference (ESM'2001)*. Vol. 9. S 185. sn. 2001, p. 65.
- [160] Eugene Y. Vasserman, Rob Jansen, James Tyra, Nicholas Hopper, and Yongdae Kim. „Membership-concealing overlay networks.“ In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. Ed. by Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis. ACM, 2009, pp. 390–399. DOI: [10.1145/1653662.1653709](https://doi.org/10.1145/1653662.1653709). URL: <http://doi.acm.org/10.1145/1653662.1653709>.
- [161] Binh Vo and Steven M. Bellovin. „Anonymous Publish-Subscribe Systems.“ In: *International Conference on Security and Privacy in Communication Networks - 10th International ICST Conference, SecureComm 2014, Beijing, China, September 24-26, 2014, Revised Selected Papers, Part I*. Ed. by Jing Tian, Jiwu Jing, and Mudhakar Srivatsa. Vol. 152. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2014, pp. 195–211. DOI: [10.1007/978-3-319-23829-6_15](https://doi.org/10.1007/978-3-319-23829-6_15). URL: http://dx.doi.org/10.1007/978-3-319-23829-6_15.
- [162] C. Wang, A. Carzaniga, D. Evans, and A. L. Wolf. „Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems.“ In: *HICSS. IEEE, 2002*, pp. 3940–3947. ISBN: 0769514359.
- [163] Tao Wang and Ian Goldberg. „Improved website fingerprinting on Tor.“ In: *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*. Ed. by Ahmad-Reza Sadeghi and Sara Foresti. ACM, 2013, pp. 201–212. DOI: [10.1145/2517840.2517851](https://doi.org/10.1145/2517840.2517851). URL: <http://doi.acm.org/10.1145/2517840.2517851>.
- [164] Masaru Watanabe and Toyotaro Suzumura. „How social network is evolving?: a preliminary study on billion-scale twitter network.“ In: *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*. Ed. by Leslie Carr, Alberto H. F. Laender, Bernadette Farias Lóscio, Irwin King, Marcus Fontoura, Denny Vrandecic, Lora Aroyo, José Palazzo M. de Oliveira, Fernanda Lima, and

- Erik Wilde. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 531–534. ISBN: 978-1-4503-2038-2. URL: <http://dl.acm.org/citation.cfm?id=2487988>.
- [165] D. J. Watts and S. H. Strogatz. „Collective dynamics of ‘small-world’ networks.“ In: *Nature* 393.6684 (1998), pp. 440–442.
- [166] Waloddi Weibull. *A statistical theory of the strength of materials*. 151. Generalstabens litografiska anstalts förlag, 1939.
- [167] Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. „The predecessor attack: An analysis of a threat to anonymous communications systems.“ In: *ACM Trans. Inf. Syst. Secur.* 7.4 (2004), pp. 489–522. DOI: [10.1145/1042031.1042032](https://doi.org/10.1145/1042031.1042032). URL: <http://doi.acm.org/10.1145/1042031.1042032>.
- [168] Renyi Xiao. „Survey on Anonymity in Unstructured Peer-to-Peer Systems.“ In: *J. Comput. Sci. Technol.* 23.4 (2008), pp. 660–671. DOI: [10.1007/s11390-008-9162-7](https://doi.org/10.1007/s11390-008-9162-7). URL: <http://dx.doi.org/10.1007/s11390-008-9162-7>.
- [169] Xiaokui Xiao and Yufei Tao. „M-invariance: towards privacy preserving re-publication of dynamic datasets.“ In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*. Ed. by Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou. ACM, 2007, pp. 689–700. DOI: [10.1145/1247480.1247556](https://doi.org/10.1145/1247480.1247556). URL: <http://doi.acm.org/10.1145/1247480.1247556>.
- [170] Zhenyun Zhuang, Yunhao Liu, Li Xiao, and Lionel M. Ni. „Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks.“ In: *32nd International Conference on Parallel Processing (ICPP 2003), 6-9 October 2003, Kaohsiung, Taiwan*. IEEE Computer Society, 2003, pp. 171–178. ISBN: 0-7695-2017-0. DOI: [10.1109/ICPP.2003.1240578](https://doi.org/10.1109/ICPP.2003.1240578). URL: <http://dx.doi.org/10.1109/ICPP.2003.1240578>.
- [171] Hubert Zimmermann. „OSI reference model–The ISO model of architecture for open systems interconnection.“ In: *Communications, IEEE Transactions on* 28.4 (1980), pp. 425–432.

COLOPHON

This document was typeset using the typographical look-and-feel classicthesis developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". classicthesis is available for both \LaTeX and \LyX :

<http://code.google.com/p/classicthesis/>

ERKLÄRUNG

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades *Dr. rer. nat.* mit dem Titel

Anonymous Publish-Subscribe Overlays

selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, 26.01.2016

Jörg Daubert